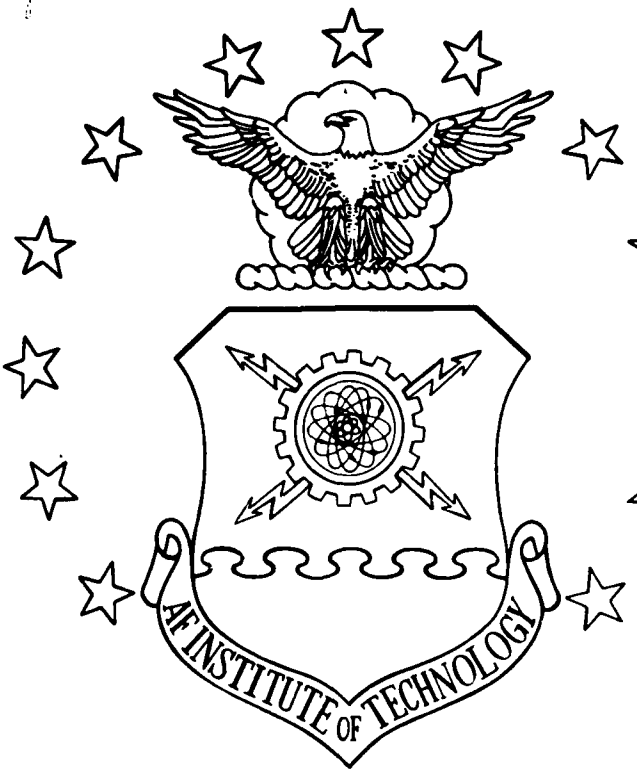

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/90M-15

COMPARISON OF BATCH MEANS AND INDEPENDENT REPLICATIONS TECHNIQUES TO APPLICATIONS OF THE KALMAN FILTER FOR SIMULATION OUTPUT ANALYSIS

THESIS

Charles H. Porter
Captain, USAF

AFIT/GOR/ENS/90M-15

AFIT/GOR/ENS/90M-15

COMPARISON OF BATCH MEANS AND INDEPENDENT REPLICATIONS TECHNIQUES TO APPLICATIONS OF THE KALMAN FILTER FOR SIMULATION OUTPUT ANALYSIS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Charles H. Porter, B.A.

Captain, USAF

March 1990

Preface

The purpose of this research was to develop software which automated the current simulation output analysis techniques of batch means and independent replications, and to determine the applicability of employing Kalman filtering for simulation output analysis on an M/M/1 queue. Both of these objectives were met, with several areas identified for follow-on research.

During the course of this thesis, many people have gone out of their way to help me. I would like to thank Major Kenneth Bauer, Dr. James Chrissis, and, especially, Dr. Peter Maybeck for being on my committee. Major Bauer's insight along with Dr. Maybeck's knowledge of the Kalman filter, and patience made many areas of investigation possible.

I would also like to thank Mr. Stan Musick, the designer of the Multimode Simulation for Optimal Filter Evaluation (MSOFE) program, and Captain Britt Snodgrass for their help in employing MSOFE. I would also like to take this opportunity to express my thankfulness to Captain Mark Gallagher for the insights and many hours of help he provided.

Finally, I wish to thank my wife, Carla, and my daughter, Megan, for their support throughout this program.

Table of Contents

List of Figures

List of Tables

Abstract

Thesis

The purpose of this ~~research~~ was twofold. First, to develop software which automated the current simulation output analysis techniques of batch means and independent replications. Second, to determine the applicability of employing Kalman filtering for simulation output analysis on an M/M/1 queue.

The Output Analysis Program incorporates batch means and independent replications into a menu driven, microcomputer program. The program allows the user to quickly analyze the simulation output graphically, discard the transient, and then perform the two techniques.

The Kalman Filter proved valuable in providing insight into the underlying system variance in steady state, and appeared to offer several avenues for further research, particularly in the area of transient identification.

COMPARISON OF BATCH MEANS AND INDEPENDENT REPLICATIONS TECHNIQUES TO APPLICATIONS OF THE KALMAN FILTER FOR SIMULATION OUTPUT ANALYSIS

I. Introduction

Aircraft scheduling and the repair of line replaceable units represent just two areas from a vast array of Air Force production processes. One problem that managers of these processes face is deciding which, if any, of the many options available, will increase their efficiency.

Computer simulation is a technique which allows managers to evaluate the benefits of several alternative courses of action quickly. In a discrete event simulation, the system to be studied is modelled as a series of events which occur at discrete points in time (22:381). The performance of the simulation model is then assumed representative of the actual system through its warm up period and into what is known as steady state.

Steady state is one of the two system states defined by Pritsker in, Introduction to Simulation and SLAM II. It has been reached when the system is oscillating about some fixed mean value. The second state is known as transient. It precedes steady state and reflects the system's "warm-up" period (22:44). Data from this state is typically removed

before most output analysis techniques are employed. The process used to discard the transient data is called, by Pritsker, "Data Truncation" and the following quote, from his book, highlights some of its limitations.

> The intent is to reduce the initial condition bias in the estimates by eliminating values recorded during the transient period of the simulation. However, by discarding a portion of the data, we are not using observations and, hence, may be increasing the estimated variance of the mean . . . The truncation point is selected as the time at which the response 'appears' to have reached steady state. (22:44)

Problems arise in deciding the point at which the system transitions from transient to near steady state, the number of simulation runs required to attain a certain degree of accuracy in the predictions, and the required length of these runs, which would ensure some representation of steady state had been achieved. There are numerous methods available (22:752-757), but each requires the analyst to make subjective decisions. It is this subjectivity that makes it possible for each of a number of different modelers to calculate their own best estimate steady state, given the same set of observation data.

There are also two main types of simulations: steady state and terminating (22:733). This research will deal with the first, where the system is assumed to have run long enough to obtain an asymptotic distribution of the target variate. In the latter, the simulation is run until a

certain event occurs or for a pre-specified length of time.

For each simulation, the number of runs required and the run length are both determined arbitrarily by the modeler. There are heuristic algorithms to aid in the determination of the number of runs required for a certain accuracy. These are based on the results of a number of "pilot runs." These values are influenced, though, by the modeler's identification and truncation of the transient. The simulation must also be allowed to run long enough to capture the extent of the transient period. For this reason, initial runs are typically much longer than necessary, and thus add to the cost of the study.

Purpose

The purpose of this thesis was twofold. The first objective was to develop a computer program which automated the batch means and independent replications techniques for simulation output analysis. The second was to determine if Kalman filtering offered any efficiencies in determining steady state characteristics of a class of queueing simulation models.

The queueing models to be evaluated will exhibit Markovian characteristics in their arrival and service rates, and will have a single activity with one server, referred to as an M/M/1 queue (24:239). The term Markovian indicates that "the conditional probability of the future

X(t+s) given the present X(s) and the past X(u), $0 \leq u < s$, depends only on the present and is independent of the past" (24:234). The arrival rate represents the average number of entities which enter the system over a specified period. In the case of this research the arrival rate is 10 entities/hr. The service rate represents the average number of entities that a particular service activity within the system can perform service on during the same time interval. For the purpose of this thesis, there is a single service activity with a mean service time of 15 entities/hr. Arrival and service rates in the M/M/1 queue are identical to birth and death rates described by Ross (24:236-237).

Kalman filtering (13:35; 16:4) will be compared to the techniques of batch means and independent replications in estimating the steady state mean and standard deviation of this estimate, as reflected by 95% confidence intervals.

In order to present the purpose of this research better, some further definition is required for the terms batch means, independent replications, Kalman filtering, time series models, and queueing simulation models.

Batch Means. The batch means procedure uses one long simulation run's output data to estimate the mean and of the system, and this estimate's variance. The output data is sectioned into sequential non-overlapping "batches" and then tested for independence between the successive batches. The

technique is described in detail by Welch (27:307), while the necessary details and equations are set forward in Chapter II.

Independent Replications. This procedure also computes estimates of the mean and its variance, but requires multiple runs to be accomplished. The mean of each run is computed and, using equations derived by Welch (27:296), the mean and its variance are computed.

Kalman Filtering. Kalman filtering (7:193; 13:35; 16:4) is a technique from the area of control theory which has among its uses the prediction of space vehicle trajectory and real time updating of inertial navigation systems. Miller and Leskiw's book, An Introduction to Kalman Filtering with Applications, provides the following explanation.

> The basic idea behind Kalman filtering is to combine the system model (the differential equation governing the flight path of the vehicle) and the measurement model (the observations made on vehicle position) in an optimum fashion in order to obtain the best estimate of the position of the vehicle at any time $t_1 > t_0$ (20:v).

Time Series Models. Time series models (1:80) predict an object's position at future discrete moments in time. They are based on a culmination of previous observations, and predict based on this history. The number of previous observations used directly in the computation of the next

position depends upon the order of the equation used. A second order model implies that the last two observed positions, and errors from their associated predictions, are combined to form the next step prediction. Chapter II addresses the foundations of the Auto Regressive - Moving Average (ARMA) model, and its application to Kalman filter generation.

Queueing Simulation Models. Queueing simulation models (22:382) are those models which evaluate a system which operates on the principle that some entities that pass through the system will have to wait in a line (queue) before they can receive some particular service. These models are used to assess the efficiency of the system in processing entities by measuring such attributes as system production, time spent in the system, and utilization rates of the specific serving functions.

Sub-objectives

In determining the effectiveness of Kalman filtering in identifying the estimate of the mean and its associated standard deviation for a system in steady state, several sub-objectives had to be accomplished. First, a computer simulation model of the M/M/1 queue had to be developed. The M/M/1 queue was selected for its well known characteristics (24:306-313) and the mathematical tractability of its steady state equations. The model was

constructed using the SLAM II ™ simulation language (22:62), with FORTRAN inserts. The output of interest from the model is the average time spent in the system for each entity.

In order to assess multiple outputs of the simulation model quickly, a micro-computer output analysis program had to be developed. This program, written in Turbo Pascal ™ (2:23), employs the batch means and independent replications techniques. The program drastically reduces the computation time, allowing the user to evaluate the effects of choosing different transient termination points quickly and easily.

The Kalman filter requires that observations be equally spaced with regard to time. This is impossible to do within a simulation with Markovian arrival/service rates. Therefore, a linear interpolation of the data was accomplished, using both system clock time and the entities' time in the system.

Using the equally spaced data, Kalman filter matrices were determined using the State Space Modeling option in the forecasting package FORECAST MASTER ™ (9:1). The matrices generated were then placed into the software package Multimode Simulation for Optimal Filter Evaluation, MSOFE (4:1), and used to determine the underlying mean and standard deviation of the system.

Overview of Chapters

Subsequent chapters will cover the thesis development. Chapter II provides the historical background and references for the techniques to be employed. It also provides derivations for the most commonly used and important equations. Chapter III, the methodology, outlines the development of the appropriate computer programs and models, and their employment for output data generation.

Chapter IV provides the results of the three output analysis techniques: batch means, independent replications, and Kalman filtering. Quantitative comparisons are based on the technique's ability to form the mean and a 95% confidence interval based on the standard deviation. Chapter V presents conclusions, founded on the results of Chapter IV, and recommendations for follow on studies.

II. Historical Development

Background

This chapter reviews applicable output analysis and Kalman filtering literature, and covers eight basic topics: steady state vs. terminating simulations, moving averages, confidence intervals, data truncation, cubic spline and linear interpolation, batch means, independent replications, and Kalman filtering.

The moving averages and data truncation techniques are used to manipulate data for use in batch means and independent replications. Moving averages "level out" the spikes in the output data by averaging across observations (27:29). Confidence intervals will be used to define a range of plausible values for the true system mean, centered on the estimated mean of the observations. The halfwidth of the confidence interval will be based upon the variance of the observations.

Data truncation refers to removing the transient or "warm-up" period data from the simulation output, leaving only steady state data (22:44). This is done to provide stationary output to be evaluated.

Recent work by Kelton (14:355-366) indicates that there could be some benefit in choosing appropriate initial system conditions. This article suggests procedures for identifying reasonable initialization parameters based on

the results of pilot runs. One of the expected benefits of the adaptive Kalman filtering procedure, however, is that once the filter has been fit, it will provide an accurate next step ahead forecast and the variance of the forecast, even with the transient data included. To this end, all simulation runs will be started with "empty and idle" conditions, but various truncation points were used when applying the batch means and independent replications techniques to evaluate the effect.

Independent replications and batch means represent two basic methods for obtaining estimates of the steady state mean and its associated variance (22:732). The main difference is that the method of independent replications requires multiple runs of the same simulation, whereas batch means uses one very long run.

These techniques are used to evaluate the output of computerized simulation models, which Welch describes in the following way:

> These models have a random input that consists of a set of sequences of random variables whose distributions are specified. . . Correspondingly, the models have a random output that consists of a set of sequences of random variables . . . whose distributions are unknown. The reason for constructing and running the simulation is to estimate certain characteristics of these output distributions. The model is simulated because these characteristics cannot be computed from analytic results. (27:268-269)

It was the purpose of this thesis to construct a computer software program which automated the batch means and independent replications procedures and to determine whether Kalman filtering techniques could be employed effectively in assessing the underlying characteristics of the system being modelled.

Steady State vs. Terminating Simulations

The differences in these two types of simulation models are as follows:

> Terminating Simulations - Simulation ends when a prespecified event or condition occurs or a certain amount of time elapses.
>
> Steady State - Assumes a situation that can be thought of as being able to run long enough that an asymptotic distribution of the target variate is (for practical purposes) realized.

This thesis dealt with the case of steady state simulations, which have reached an asymptotic approximation of true steady state behavior. It is important to note that true steady state is impossible to achieve in a process defined on a finite time interval.

Moving Averages

This technique can be used to remove large fluctuations from the graphical portrayal of the output data. Performing a moving average (27:293-294) provides a new "averaged" value ($\hat{\mu}_n$) for each of the N original data points. These

values are then plotted, and reviewed to determine trends in the true system mean, μ. This aids the experimenter by making it easier to interpret the data to determine the extent of the transient period (27:293), for M runs of the simulation. Welch addresses the technique in the following way:

> . . . sometimes these judgments about the long term trends in $\{\mu_n\}$ are easier to make if an explicit attempt is made to smooth out the short term (high frequency) fluctuations in $\{\hat{\mu}_n\}$. The simplest way to do this is to take a moving average over an interval long enough to remove short term fluctuations but not so long as to distort the long term trend. (27:293-294)

A moving average of length 2K + 1 is defined as

$$\hat{\mu}(n;K) = \begin{cases} (2K+1)^{-1} \sum_{k=-K}^{K} \hat{\mu}_{n+k} & \text{if } n \geq K+1 \\ (2n-1)^{-1} \sum_{k=-(n-1)}^{n-1} \hat{\mu}_{n+k} & \text{if } n < K+1 \end{cases} \quad (1)$$

The next two figures present a graphical comparison of the two techniques. Figure 1 depicts a representative computer simulation output with customer time in system represented on the Y-axis and the number of the entity on the X-axis. Figure 2 shows the data from Figure 1, but with moving average values from a window with a K value of 5.

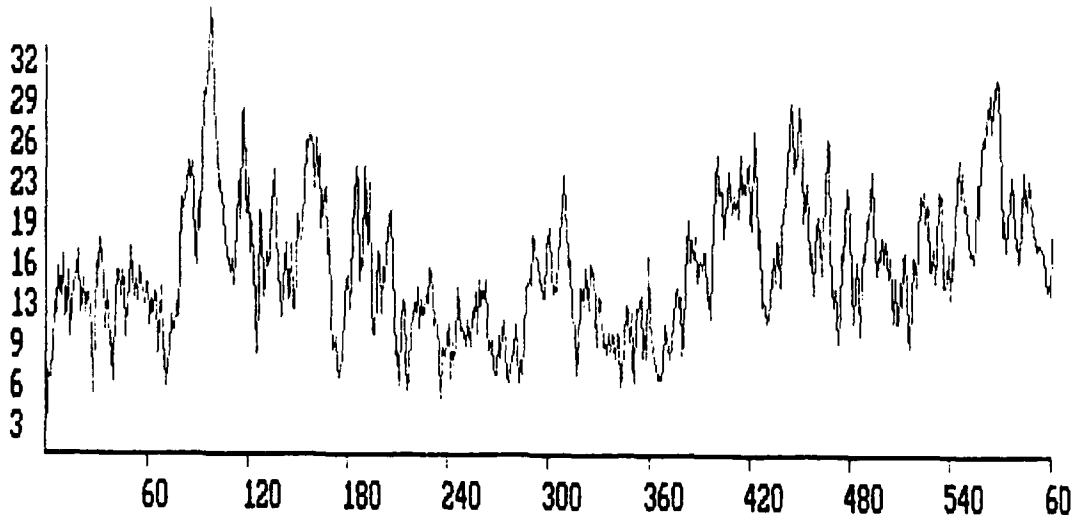

Figure 1. Representative Simulation Model Output Data

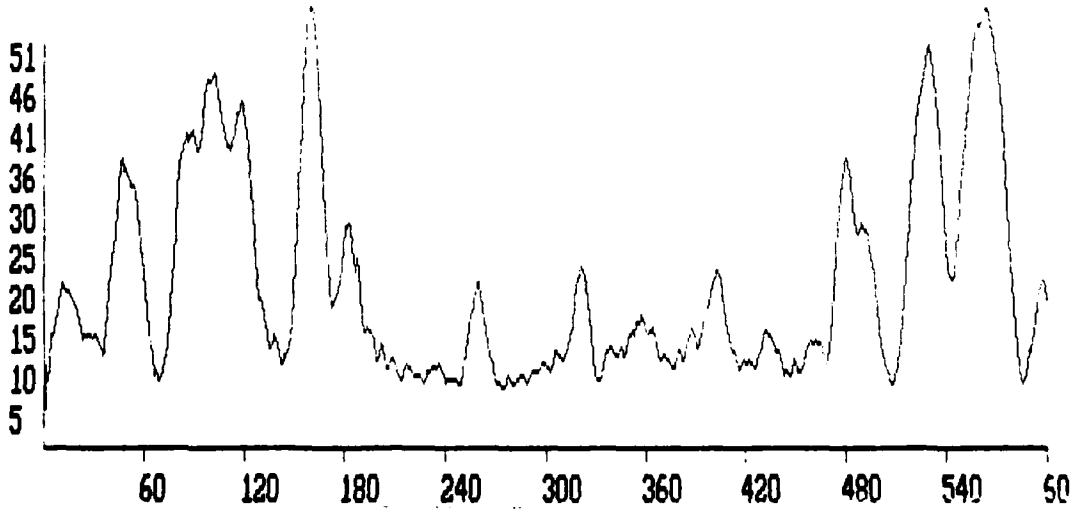

Figure 2. Moving Averaged Simulation Output.

Confidence Intervals

Confidence intervals are used when it isn't possible to arrive at an exact answer. The intervals are centered at the best estimate attainable, and then a "halfwidth" value is subtracted and added to form the interval. The halfwidth is determined by three factors: the number of observations to be used, the significance level (usually denoted by α), and the standard deviation of the estimate (6:266-268). See Equation (6) below for the actual computation (6:268).

Equation (6) provides a halfwidth which will statistically cover the mean $(1-\alpha)$% of the time. This is because the t value is derived from the fact that the population from which the mean was estimated is assumed to have a normal distribution. In most cases the true variance of the system is unknown and so the standard deviation (σ) is approximated by the standard deviation of the estimates (s).

In general, when estimating confidence intervals, the equations below are used. In keeping with Welch's notation the number of runs is represented by "M", the number of observations per run by "N,", and the n^{th} observation of the m^{th} run by V_{mn} (27:290).

$$V_{1n}, \qquad n = 1,\ldots,N,$$

$$V_{2n}, \qquad n = 1,\ldots,N,$$

$$\vdots$$

$$V_{Mn}, \qquad n = 1,\ldots,N, \tag{2}$$

Then the mean of the m^{th} run can be computed as:

$$\hat{\mu}_m = \bar{V}_m = 1/M \sum_{n=1}^{N} V_{mn}, \qquad m = 1,\ldots,M, \tag{3}$$

Using the mean values for each run, the variance and standard deviation can be determined using the following equations (27:294).

$$Var_m = s_m^2 = 1/(M-1) \sum_{n=1}^{N} (V_{mn} - \hat{\mu}_m)^2 \tag{4}$$

$$Std\ Dev_n = s_n = (var_n)^{1/2} \tag{5}$$

These values are then combined with the t critical value to determine the boundaries of the confidence interval using the equation below, where $(1 - \alpha)$ represents the confidence coefficient and $(n - 1)$ indicates the degrees of freedom for the t value (6:268).

$$Halfwidth = x \pm t_{\alpha/2,n-1} * s/\sqrt{n} \tag{6}$$

Data Truncation

This technique eliminates the transient period data within a run. In this application, the cutoff value is determined by the experimenter, based on a graphical

display, as shown above in Figure 1.

Once the cutoff value, n_0 , is established, all observations previous to this point (from 1 to n_0) are deleted across the M runs. This leaves the data which represents the system's performance in steady state (27:289).

As mentioned previously, Pritsker highlights some limitations with this technique. "The truncation point is selected as the time at which the response 'appears' to have reached steady state" (22:44).

This arbitrary selection can lead to various problems as noted by Kleijnen:

> Practitioners often throw away the [initial] part of the time series; that is, they "warm-up" the simulation before they start recording observations. Unfortunately, two practical problems remain.
>
> 1. How can we determine whether the transient phase is over?
> 2. Throwing away the initial phase of each run wastes computer time.
>
> Practitioners often construct graphs (and making graphs is always an excellent idea in any statistical experiment) to see whether start-up effects have "obviously" disappeared. . . If we overestimate the length of the transient phase, we throw away information on the steady state and increase the variance of the final estimator. If we do not wish to waste computer time, we may be tempted to underestimate the initial phase and we bias the final estimator. (15:65)

Cubic Spline and Linear Interpolation

The simulation output data provided by the SLAM II model is unequally spaced in time. This is caused by the exponential nature of the arrival and service rates. In order to employ the adaptive Kalman filtering program FORECAST MASTER, however, the input data is assumed to be at regular time intervals.

Two methods were identified to transform the output data into these regular intervals. The first, cubic spline interpolation, fits a continuous function through the data points. This function passes through each of the data points and is continuously differentiable. Burden and Faires (3:118) provide the following explanation:

> A general cubic polynomial involves four constants; so there is sufficient flexibility in the cubic spline procedure to ensure not only that the interpolant is continuously differentiable on the interval, but also that it has a continuous second derivative on the interval. The construction of the cubic spline does not, however, assume that the derivatives of the interpolant agree with those of the function, even at the nodes.

Burden and Faires also provide a derivation of the equations used to form the cubic spline (3:118-120), along with several computer algorithms for different applications. The program MATRIX$_x$ ™ (12:4-34) was used to obtain the cubic spline for the unequally spaced output data.

The second method used was that of piecewise linear

interpolation. This procedure connects adjacent data values by straight lines, and then computes equally spaced observations. In this experiment a regular time interval of 10 minutes was used. Appendix D contains a copy of the program used to obtain the linear data and some sample output. Data points were interpolated for the first 1000 data points in each of the 50 output data files.

Independent Replications

Also called "replications," this technique deals with multiple runs of the same simulation, assuming independence between the runs. The method of replications calculates an estimate of the mean and its corresponding variance (22:732).

After the transient data has been removed from each of the M runs, the remaining values all have approximately the same mean, which can be computed with the following equations (27:296).

> In the method of independent replications, independent sample means are generated from independent simulation runs. A point estimate and a confidence interval are generated from these sample means as we shall now describe . . .
>
> On each replication we form the sample mean
>
> $$\hat{\mu}_m = 1/(N-n_0) \sum_{n=n_0+1}^{N} V_{mn} \qquad (7)$$
>
> . . . approximately unbiased estimator of μ is

$$\hat{\mu} = 1/M \sum_{m=1}^{M} \hat{\mu}_m \qquad (8)$$

Moreover, if we let

$$s^2(\hat{\mu}_m) = 1/(M-1) \sum_{m=1}^{M} (\hat{\mu}_m - \hat{\mu})^2 = 1/(M-1) \sum_{m=1}^{M} \hat{\mu}_m^2 - M/(M-1)\,\hat{\mu}^2 \qquad (9)$$

. . . we obtain the confidence interval and statement

$$\text{Prob}\{\hat{\mu} - t_{M-1}(1-\alpha/2)(s(\hat{\mu}_m)/M^{1/2} \le \mu \le \hat{\mu} + t_{M-1}(1-\alpha/2)s(\hat{\mu}_m)/M^{1/2}\} = 1-\alpha \qquad (10)$$

According to Pritsker the technique has the following advantages and disadvantages.

> The replication procedure has the desirable property that samples are independent. Another advantage is that it can be used for both terminating and steady state analysis where a terminating analysis is one that is performed for a specific finite time period. The disadvantages associated with replications are: 1) each replication contains a startup segment which may not be representative of stationary behavior; and 2) only one sample, X_i, is obtained from each replication which could mean that extensive information about the variable of interest is not being gleaned from the data. (22:733)

Batch Means

This technique deals with the case of using only one long run of output data. According to Welch it "exactly parallels the method of independent replications except that

the sequences are adjacent, non-overlapping subsequences of the output of a single simulation run" (27:307).

The most important part is to determine the batch size, m, necessary to indicate independence among the batches (8:511). Once this independence is identified, the batches can be compared as if they were separate runs (5:303).

Chen and Seila state the ". . . batch means method depends . . . upon establishing a batch size that is large enough that all batches are approximately uncorrelated" (5:302). A method which tests this autocorrelation between the batches was developed by Fishman, and uses a test statistic C_k (8:514). When the autocorrelation effect is larger than the C_k statistic, it indicates that independence between the batches is reasonable.

Fishman's work uses the symbol Y to represent individual observations. To keep with the notation used earlier in the text, the following equations will have the constant V in place of Y. Within the equation, k is the total number of batches, and m is the number of observations per batch (8:514):

$$C_k = 1 - \sum_{i=1}^{k-1} (V_{i,m} - V_{i+1,m})^2 / 2 \sum_{i=1}^{k} (V_{i,m} - \mu_n)^2 \qquad (11)$$

The algorithm starts by setting the number of batches

equal to the number of observations. It then determines values for the test statistic. If the values indicate independence, then the testing stops and a mean and confidence interval are computed. If the values do not pass the test, then the number of batches is divided by two (k/2), resulting in a doubled batch size. The autocorrelation and test statistic are then compared, and the process continues until the values pass the independence test or the number of batches becomes less than eight (8:515). In explaining why at least eight batches are required, Fishman references Von Neumann's work, stating a comparison at the .05 significance level for C_k with k >= 8 "indicates negligible error" (8:514).

Chen and Seila identify the following equations for computing the mean for the data from the independent batches. The constant "i" represents the number of the batch, while k is the number of batches (5:302).

The i^{th} batch mean, then is

$$\bar{X}_i = 1/k \sum_{j=1}^{k} X_{(i-1)k+j} \qquad (12)$$

. . . the point estimator for μ is the sample mean of the batch means

$$\hat{\mu} = 1/m \sum_{i=1}^{m} \bar{X}_i \qquad (13)$$

The standard deviation of the estimator and its confidence interval can be calculated with Eqs. (9) and (10) above by substituting m for M.

Pritsker refers to this technique as "subintervals" and highlights the following advantages and disadvantages.

> The advantages of using subintervals to estimate the variance of the sample mean are that a single run can be used to obtain an estimate and only one transient period is included in the output (or required to be deleted). The disadvantage of the procedure is in establishing the batch size, b, which makes the subintervals independent. (22:735)

Kalman filtering

Kalman filtering is described by Maybeck (16:4) as an "optimal recursive data processing algorithm." In his text, Stochastic Models Estimation, and Control, he describes the Kalman filter's underlying assumptions (16:7).

> A Kalman filter performs this conditional probability density propagation for problems in which the system can be described through a linear model in which system and measurement noises are white and Gaussian (to be explained shortly). Under these conditions, the mean, mode, median, and virtually any reasonable choice for an "optimal" estimate all coincide, so there is in fact a unique "best" estimate of the value of x.

According to Maybeck,"whiteness implies that the noise value

is not correlated in time," and being Gaussian implies that, "at any single point in time, the probability density of a Gaussian noise amplitude takes on the shape of a normal bell shaped curve." One of the greatest strengths of the Kalman filter is its versatility once these three assumptions are made. It can be implemented over a wide spectrum of applications from many fields.

Recent work by the U.S. Army at Fort Monmouth, New Jersey (26:9-32), applied Kalman filtering to estimating "clocks." This study indicated the benefit of Kalman filtering, comparing it to standard auto regressive integrated moving average (ARIMA) models.

Static Case. In an article entitled Understanding the Kalman Filter, Meinhold and Singpurwalla highlight the fact that most of the documentation in this area has been done by the engineering and scientific community. Therefore, the notation and methodology are typically difficult to relate with common statistical formulae (19:123). They have taken the basic concepts of the Kalman filter and, using Bayesian formulation and multivariate statistics, have provided a statistical derivation of this important process in statistical terms.

Meinhold and Singpurwalla provide a clear path to follow. They develop the concept of the Kalman filter as the combination of system and observation equations.

The following quote identifies the components of the system/observation equations and their significance:

> Let Y_t , Y_{t-1} ,..., Y_1 , the data (which may be either scalar or vectors), denote the observed values of a variable of interest at times t, t-1,..., 1. We assume that Y_t depends on an unobservable quantity θ_t , known as the <u>state of nature</u>. Our aim is to make inferences about θ_t , which may be either a scalar or a vector and whose dimension is independent of the dimension of Y_t . The relationship between Y_t and θ_t is linear and is specified by the <u>observation equation</u>
>
> $$Y_t = F_t \theta_t + v_t \qquad (14)$$
>
> where F_t is a known quantity. The <u>observation error v_t</u> is assumed to be normally distributed with mean zero and a known variance V_t , denoted as $v_t \sim N(0, V_t)$.
>
> The essential difference between the [Kalman filter] and the conventional linear model representation is that in the former, the state of nature - analogous to the regression coefficients of the latter - is not assumed to be a constant but may change with time. This dynamic feature is incorporated via the <u>system equation</u>, wherein
>
> $$\theta_t = G_t\theta_{t-1} + w_t \qquad (15)$$
>
> G_t being a known quantity, and the <u>system equation error</u> $w_t \sim N(0, W_t)$, with W_t known...
>
> In addition to the usual linear model assumptions regarding the error terms, we also postulate that v_t is independent of w_t .

To provide an example, Meinhold and Singpurwalla

present the tracking of a satellite's orbit (19:124). This provides some insight as to the physical relationships within the equations.

> The unknown state of nature θ_t could be the position and speed of the satellite at time t, with respect to a spherical coordinate system with origin at the center of the earth. These quantities can not be measured directly. Instead, from tracking stations around the earth, we may obtain measurements of distance to the satellite and the accompanying angles of measurement; these are the Y_t's. The principles of geometry, mapping Y_t into θ_t, would be incorporated in F_t, while v_t would reflect the measurement error; G_t would prescribe how the position and speed change in time according to physical laws governing orbiting bodies, while w_t would allow for deviations from these laws owing to such factors as nonuniformity of the earths gravitational field, and so on.

The key to the Kalman filtering algorithm lies in its "recursive" nature. Future predictions are based only on the last state and its corresponding, cumulative covariance kernel, a matrix containing the covariance estimates for the states of the process. The process is completely defined by its first two moments, due to the underlying assumption that the process can be represented by a linear system driven by white, Gaussian (normal) noise.

The main intent of Meinhold and Singpurwalla's article is to derive inferences about θ_t, given the input values Y, using Bayesian statistics. The following will overview the derivation of the Kalman filtering equations, while expanding on some of the actual calculations.

The derivation through the direct application of Bayes theorem:

$$\text{Prob}(\text{State of Nature} \mid \text{Data}) \propto \text{Prob}(\text{Data} \mid \text{State of Nature}) \times \text{Prob}(\text{State of Nature})$$

or

$$P(\theta_t \mid Y_t) \propto P(Y_t \mid \theta_t, Y_{t-1}) \times P(\theta_t \mid Y_{t-1}) \qquad (16)$$

where

$$Y_{t-1} = (Y_{t-1}, Y_{t-2}, \ldots, Y_1)$$

As noted by Meinhold and Singpurwalla, the left side of the equation represents the <u>posterior distribution</u> for θ at time t, while the terms on the right hand side represent the <u>likelihood</u> and <u>prior distributions</u> respectively.

At this point, knowledge of the value of θ_{t-1} can be stated in a probability statement, where $\hat{\theta}$ is the best estimate of the system state at the last time, and Σ is the associated covariance kernel of this estimate:

$$(\theta_{t-1} \mid Y_{t-1}) \approx N(\hat{\theta}_{t-1}, \Sigma_{t-1}) \qquad (17)$$

The recursive estimation begins at time 0, with initial estimations θ_0 and Σ_0. The process then looks forward to system performance a time t in the future. This is done in two increments: the first is a propagation of the initial estimates, the second an update based on the actual

observation.

In the propagation phase of the estimation, the best choice for the value of θ_t is given by the system equation $G_t\theta_{t-1} + w_t$. Based on the well known statistical relationship:

$$X \approx N(\mu,\Sigma) \Rightarrow CX \approx N(C\mu, C\Sigma C') \qquad (18)$$

the mean and variance of θ_t can be seen to be

$$(\theta_t | Y_{t-1}) \approx N(G_t\hat{\theta}_{t-1}, R_t = G_t\Sigma_{t-1}G_t' + W_t) \qquad (19)$$

which represents the prior distribution.

Upon making the observation at time t, the posterior probabilities must now be computed. This requires the estimation of the likelihood $L(\theta_t | Y_t)$, or equivalently $P(Y_t | \theta_t, Y_{t-1})$. This estimation follows, where e_t represents the error in predicting Y_t from the point t-1, also known as the residual:

$$e_t = Y_t - \hat{Y}_t = Y_t - F_t G_t\hat{\theta}_{t-1} \qquad (20)$$

As F_t, G_t, and $\hat{\theta}_{t-1}$ are all known, knowing Y_t is equivalent to knowing e_t. With this information, the probability statement is now rewritten:

$$P(\theta_t | Y_t, Y_{t-1}) = P(\theta_t | e_t, Y_{t-1}) \propto P(e_t | \theta_t, Y_{t-1}) \times P(\theta_t | Y_{t-1}) \qquad (21)$$

with $P(e_t|\theta_t, Y_{t-1})$ being the likelihood.

Now, incorporating the observation to replace Y_t in the equation for e_t above, the result is show to be

$$e_t = F_t(\theta_t - G_t\hat{\theta}_{t-1}) + v_t \tag{22}$$

making the value of $E(e_t|\theta_t, Y_{t-1}) = F_t(\theta_t - G_t\hat{\theta}_{t-1})$.

As the distribution of v_t has been assumed to be $N(0,V_t)$, it can be seen that the likelihood can be described as follows:

$$(e_t|\theta_t, Y_{t-1}) \approx N(F_t(\theta_t - G_t\hat{\theta}_{t-1}), V_t) \tag{23}$$

Bayes' theorem now results in

$$P(\theta_t|Y_t, Y_{t-1}) = \frac{P(e_t|\theta_t, Y_{t-1}) \times P(\theta_t|Y_{t-1})}{\int P(e_t, \theta_t|Y_{t-1})d\theta_t} \tag{24}$$

This is the best estimator of θ_t at time t. Once this value is computed, the process is repeated for a point, time t in the future.

Meinhold and Singpurwalla note (19:125) that the effort to determine $P(\theta_t|Y_t)$ above could be reduced by using well known results from multivariate statistics.

> Let X_1 and X_2 have a bivariate normal distribution with means μ_1 and μ_2, respectively, and a covariance matrix
>
> We denote this by $\begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$ (25)

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N \left[\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right] \tag{26}$$

When [26] holds, the conditional distribution of X_1 given X_2 is described by

$$(X_1 | X_2 = x_2) \approx N(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2), \Sigma_{11}-\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}) \tag{27}$$

The quantity $\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$ is called the <u>regression function</u>, and $\Sigma_{12}\Sigma_{22}^{-1}$ is referred to as the <u>coefficient of the least squares regression of X_1 on x_2</u> ...

For our situation, we suppress the conditioning variables Y_{t-1} and let X_1 correspond to e_t, and X_2 correspond to θ_t ; we denote this correspondence by $X_1 <=> e_t$ and $X_2 <=> \theta_t$. Since

$$(\theta_t | Y_{t-1}) \approx N(G_t\hat{\theta}_{t-1}, R_t) \text{ (see(19))}$$

we note that

$$\mu_2 <=> G_t\hat{\theta}_{t-1}$$

and

$$\Sigma_{22} <=> R_t$$

If in (26) we replace X_1, X_2, μ_2, and Σ_{22} by e_t, θ_t, $G_t\theta_{t-1}$, and R_t, respectively and recall the result that

$$(e_t | \theta_t, Y_{t-1}) \approx N(F_t(\theta_t - G_t\hat{\theta}_{t-1}), V_t). \tag{28}$$

With [28] and the fact that

$$\theta_t \approx N(G_t\hat{\theta}_{t-1}, R_t)$$

the expected value for (27) can be computed as

$$E(X_1 | X_2 = x_2) = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \tag{29}$$

where, substituting in for X_1, X_2, μ_2, and Σ_{22}

$$E(e_t | \theta_t, Y_{t-1}) = \mu_1 + \Sigma_{12} R_t^{-1} (\theta_t - G_t \hat{\theta}_{t-1}). \qquad (30)$$

Comparing this to (28) results in

$$\mu_1 + \Sigma_{12} R_t^{-1} (\theta_t - G_t\theta_{t-1}) = F_t(\theta_t - G_t\theta_{t-1}), \qquad (31)$$

which implies that $\mu_1 = 0$, and $\Sigma_{12} = F_t R_t$.

With this information, the variance of this function is computed as follows, where R_t is symmetric:

$$\begin{aligned} Var(e_t | \theta_t, Y_{t-1}) &= \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} \qquad (32) \\ &= \Sigma_{11} - F_t R_t R_t^{-1} R_t F_t' \\ &= \Sigma_{11} - F_t R_t F_t' \end{aligned}$$

From (28), this variance equals V_t, therefore

$$V_t = \Sigma_{11} - F_t R_t F_t' \qquad (33)$$

or, equivalently

$$\Sigma_{11} = V_t + F_t R_t F_t' \qquad (34)$$

Substituting these results into (26) gives

$$\begin{pmatrix} \theta_t | Y_{t-1} \\ e_t | Y_{t-1} \end{pmatrix} \approx N\left[\begin{pmatrix} G_t \theta_{t-1} \\ 0 \end{pmatrix}, \begin{bmatrix} R_t & R_t F_t' \\ F_t R_t & V_t + F_t R_t F_t' \end{bmatrix}\right]$$

Now substitute into

$$(\theta_t | e_t, Y_{t-1}) \approx N(\mu_1 + \Sigma_{12} \Sigma_{22}^{-1} (x_2 - \mu_2), \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}) \qquad (35)$$

and derive the following expressions for mean, θ_t, and

variance, Σ_t.

$$\hat{\theta}_t = E(\theta_t | e_t) = G_t \hat{\theta}_{t-1} + R_t F_t' (V_t + F_t R_t F_t')^{-1} e_t \quad (36)$$

$$\Sigma_t = Var(\theta_t | e_t) = R_t - R_t F_t' (V_t + F_t R_t F_t')^{-1} F_t R_t \quad (37)$$

where $R_t F_t' (V_t + F_t R_t F_t')^{-1}$ represents the Kalman filter gain, K.

Now that the values for θ_t and Σ_t have been derived, all that is required is to specify a system's G and F matrices and the initial values for θ_0 and Σ_0. The program FORECAST MASTER has the same type of Kalman filtering approach, but uses the "innovations form" of the Kalman filter (9:7-6), as shown on page 34.

Adaptive Filtering. Adaptive Kalman filtering falls typically into one of two styles. The first incorporates a bank of Kalman filters which are each built for a particular type of system performance, such as steady state, a transient value, or a step change. Harrison and Stevens (10:343-344) apply this strategy in short term forecasting, using a Bayesian approach.

The filters' outputs are then fed into a controller which evaluates the residual values from each filter. The controller then selects the best filter based on the lowest residual magnitude, relative to the filter-computed residual covariance $[V_t + F_t R_t F_t']$. This process is repeated at each

covariance $[V_t + F_t R_t F_t']$. This process is repeated at each input value, allowing the system to react instantly to state changes in the data.

The second style, and the one used in this thesis, depends upon a single Kalman gain matrix, generated by maximum likelihood estimations. The process to be used stems from the work of Mehra (18:175-184), and uses maximum likelihood methods to reduce the residual values to the level of white noise. This was done by fitting the ARMA model which had the residual values most in consonance with the filter-computed covariance $[V_t + F_t R_t F_t']$, and then representing this model in state space form.

ARMA models combine the aspects of two different types of times series models, autoregressive and moving average, to obtain the most parsimonious representation of the system. The autoregressive (AR) model is based on the findings of Yule and Wold, as referenced by Abraham and Ledolter. The process is based on the observation that "every weakly stationary nondeterministic stochastic process $(z_t - \mu)$ can be written as a linear combination (or linear filter) of a sequence of uncorrelated random variables." (1:197) The order of the model determines the number of terms required. For the AR model, the order is typically denoted by the letter "p." Equation (38) shows the general equation for an AR(p) process:

$$z_t = \phi_1 z_{t-1} + \cdots \phi_p z_{t-p} + a_t \qquad (38)$$

where the value a_t represents the random shock at time t, from "a sequence of uncorrelated random variables from a fixed distribution with mean $E(a_t) = 0$, and variance $V(a_t) = \sigma^2$." (1:197) The values of the ϕ_i expressions represent the coefficients associated with the previous p observations.

The moving average (MA) model is based on using the previous "random shock" values, a_i, in conjunction with their appropriate weighting coefficients. These coefficients, are usually denoted as θ_i, but, to avoid confusion with those of the steady state Kalman filter, the variable τ_i will be used instead. Equation (39) represents the general MA (q) model, where "q" denotes the model's order (1:217):

$$z_t - \mu = a_t - \tau_1 a_{t-1} - \cdots - \tau_q a_{t-q} \qquad (39)$$

Although either of these techniques can be used to model a system, the most parsimonious representation is usually a combination of the two, in the form of the ARMA model, where a "backshift operator is employed. This operator, "B," denotes stepping backward through time. In general $Bz_t = z_{t-1}$, and $B^a z_t = z_{t-a}$. The general ARMA (p,q) model is shown below in equation 40 (1:222).

$$(1 - \Xi_1 B - \ldots - \Xi_p B^p)(z_t - \mu) =$$

$$(1 - \tau_1 B - \ldots - \tau_q B^q)\, a_t \qquad (40)$$

Once the optimal ARMA model has been derived, it is placed into a state space form. For details on this procedure see Harvey (11:101-104), Shea (25:92), or Abraham and Ledolter (1:359-360). Abraham and Ledolter use S_t to represent the system states, for which θ_t will be substituted to keep with previous notation. Looking at their general case for an ARMA (p,q) model (1:360), where [G] is an identity matrix and there is no measurement noise:

$$y_t = \Xi_1 + \ldots + \Xi_p y_{t-p} + a_t - \tau_1 a_{t-1} - \ldots - \tau_q a_{t-q} \qquad (41)$$

the state model is given by

$$y_t = [1\ 0\ 0\ 0\ \ldots\ 0]\theta_t$$

$$\begin{bmatrix} \theta_{t+1,1} \\ \theta_{t+1,2} \\ \cdot \\ \cdot \\ \theta_{t+1,k} \end{bmatrix} = \begin{bmatrix} \Xi_1 & & \\ \Xi_2 & I_{k-1} & \\ \cdot & & \\ \cdot & & \\ \Xi_k & 0\ \ldots\ 0 & \end{bmatrix} \begin{bmatrix} \theta_{t,1} \\ \theta_{t,2} \\ \\ \\ \theta_{t,k} \end{bmatrix} + \begin{bmatrix} 1 \\ -\tau_1 \\ \cdot \\ \cdot \\ -\tau_k \end{bmatrix} \bullet a_{t+1} \qquad (42)$$

The program FORECAST MASTER then used these values to form the appropriate Kalman filter matrices for computation, using the Goodrich-Larimore algorithm (9:7-8), a variation of the Akaike-Mehra algorithm, to identify the best values of state transition G_t, Kalman gain K, and measurement (forecast) matrix F_t of Eqs. (36) and (37). The format used

by FORECAST MASTER, however, was the algebraically equivalent "innovations form" of the Kalman filter (9:7-6). This form is similar to the equations derived above, but results in the following equations:

$$\hat{\theta}_t = G_t \hat{\theta}_{t-1} + [G_t K] v_t$$

$$Y_t = F_t \hat{\theta}_t + S_t v_t$$

where

$$S_t v_t = Y_t - F_t \hat{\theta}_t = r_t$$

$$S_t = [V_t + F_t R_t F_t']^{1/2}$$

$$v_t = S_t^{-1} [Y_t - F_t \hat{\theta}_t]$$

$$[G_t K]_t = G_t R_t F_t' (S_t')^{-1}$$

resulting in the equations:

$$\hat{\theta}_t = G_t \hat{\theta}_{t-1} + [G_t K]_t S_t^{-1} r_t$$

and

$$Y_t = F_t \hat{\theta}_t + r_t$$

Summary

There are several techniques currently available to evaluate the mean and variance of simulation output. The benefits of these procedures include their mathematical tractability and their easy application of classical statistics. Drawbacks include the confusion on the length

of the transient, and coverage rate of the system mean given bias.

III. Methodology

Overview

This chapter will cover the development of the methods used for this thesis. In generating the computer software package which automated the output analysis techniques of batch means and independent replications, the intent was to provide the analyst with a quick, easy-to-use analytic tool. This program was then employed to evaluate the possible benefits of Kalman filtering for performing the same tasks. The measure of effectiveness was each technique's ability to calculate the mean and confidence interval halfwidth, based on the standard deviation. The impact of choosing the cutoff value for the transient period was also addressed.

There are four main topic areas in this chapter: model formulation, output analysis programming, the application of Kalman filtering, and data formatting.

Model formulation

In order to compare the different techniques, an M/M/1 simulation queueing model was developed using SLAM II. Figure 3 shows a flowchart of the M/M/1 queue, while the SLAM II and FORTRAN insert programs are contained in Appendix A.

Looking at the individual components in Figure 3, there are four distinct system "nodes" and one service activity.

The first node encountered is a create node, which generates each of the entities which will pass through the system, at a specified "arrival rate." In this research the arrival rate has been selected to be exponential, averaging 10 entities per hour. This results in an entity being created, on the average, every 6 minutes. The entity then proceeds to an infinite capacity queue node, where it waits for the service activity to be free. The waiting criteria for this node is First-In-First-Out (FIFO). As the service activity, located between the queue and event nodes, becomes free, the next entity in the queue, if there is one, enters the activity for service. Service activities can have single or multiple "servers" which perform the service. In this instance there is only one server, and so all entities must pass through the activity one at a time. The "service rate" for the server represents the average production over time. In this case, the service rate is exponential with a mean value of 15 entities/hr.

Upon completion of the activity, the entity enters the event node. At this point, the FORTRAN subroutine is accessed to build the file containing the actual system time (TNOW) and the time the entity has spent in the system (TNOW - Time of Creation). After this is accomplished the entity is removed from the system by the terminate node.

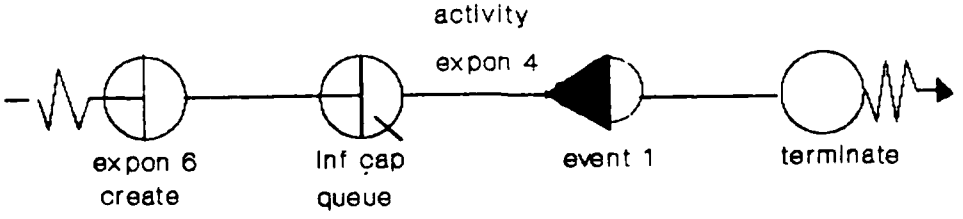

Figure 3. Flowchart of M/M/1 Queueing Simulation

The model was run 50 times, with four replications per run, and 1000 entities per replication. The data structure is shown in Figure 4, where the first column of 4000 values represents system clock time. This column was only required for the Kalman filtering technique, and is discussed below. The second column provides the observations for the actual time in system for each entity, and was used by all of the techniques.

Output Analysis Programming

The SLAM II model output consisted of 400,000 data values. As can be imagined, performing the batch means and independent replications techniques manually on this

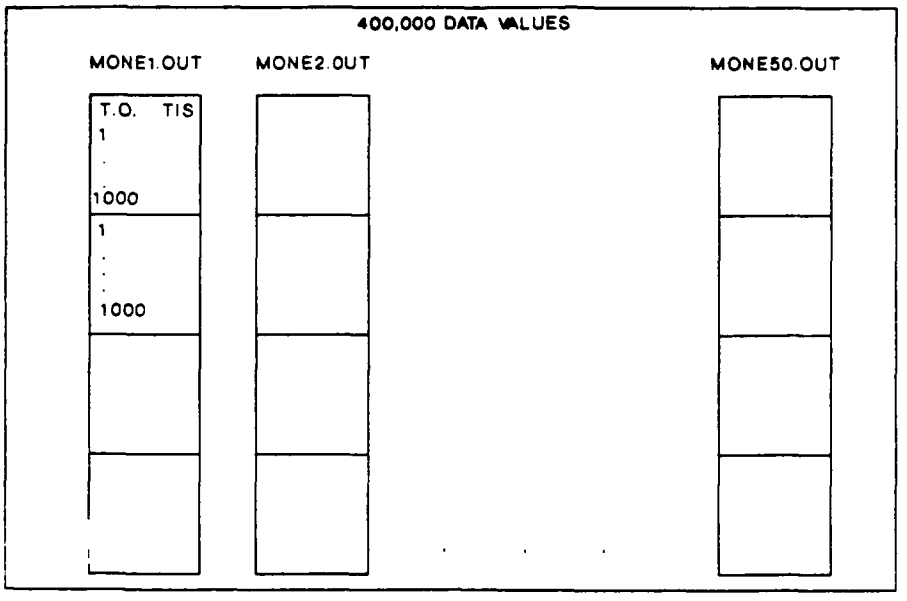

Figure 4. Configuration of SLAM II Output

quantity of data is untenable. The Output Analysis Program was, therefore, developed to process the data quickly on a micro-computer. The program was written in Turbo Pascal (2:1), and is shown along with its corresponding user's manual in Appendices B and C.

The program quickly averages the data, presenting it to the analyst in a graphic display, from which to determine the extent of the transient. Figure 5 is a wiring diagram of the program, indicating the relationships between each of the individual procedures.

Each of the procedures will now be discussed briefly. The Data Menu allows the user to build a new data directory, recall an existing directory, or list all directories made previously that are available. Throughout the thesis, the

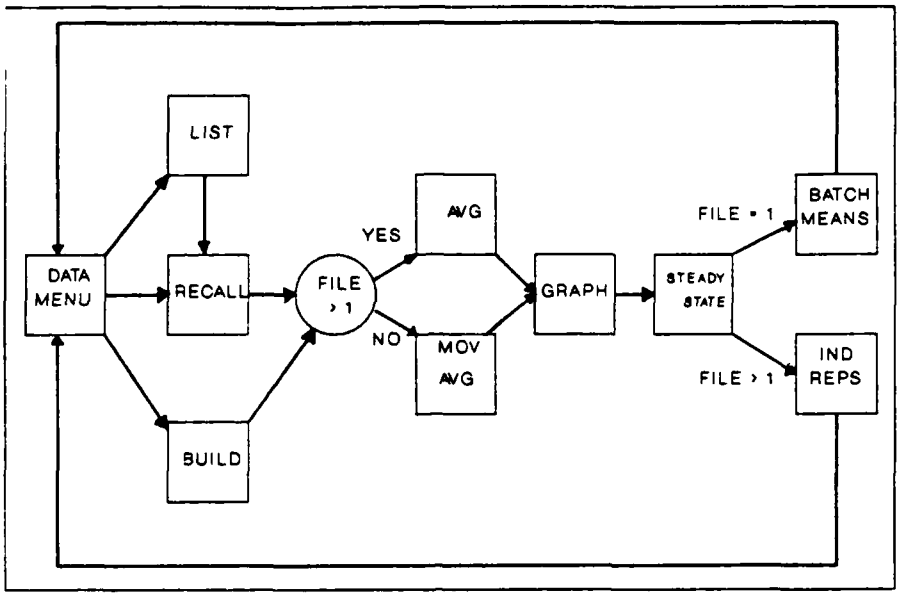

Figure 5. Flowchart of Output Analysis Program

term directory will be used to denote a string of SLAM II output files sequentially listed. The data directories are built by reading in a column of sequential output files of uniform length. For example, in the case of the thesis, four output files of length 1000 are read in. The user is asked to provide a name for the directory. The program then takes this name, adds a .DCT extension, and saves the file to disk for future use. The recall option simply retrieves one of these previously built directories, while the list option provides the user a listing of .DCT files that currently exist.

Once the data directory has been built or recalled, the user is asked whether to evaluate multiple or single output

files. If the multiple file option is selected, the program averages each data point across all replications and plots these averages on the screen. For instance, the first observation of each of the four runs would be added together to obtain a total, this total divided by 4 would then be plotted at x = 1. This procedure was repeated 1000 times.

When the single file option was selected, the program used the technique of moving averages to smooth out the data. The equations and methodology for this technique are covered in Chapter II. The user must specify the moving average window's halfwidth. A plot of the averaged data is then presented on the screen.

When the data from either option is plotted on the screen, the user has the ability to scroll forwards or backwards through the data, 600 points at a time. The user then determines an estimate of where the transient period ends, and enters this value. The program takes this in as the variable CUTOFF. An example of the final screen display is shown in Figure 6.

With the CUTOFF value identified, the program goes through each of the output files in the directory and truncates the first CUTOFF number of values. The user is asked to name the new file containing only the steady state data. The name specified is given a .SST extension by the program and is saved to the disk.

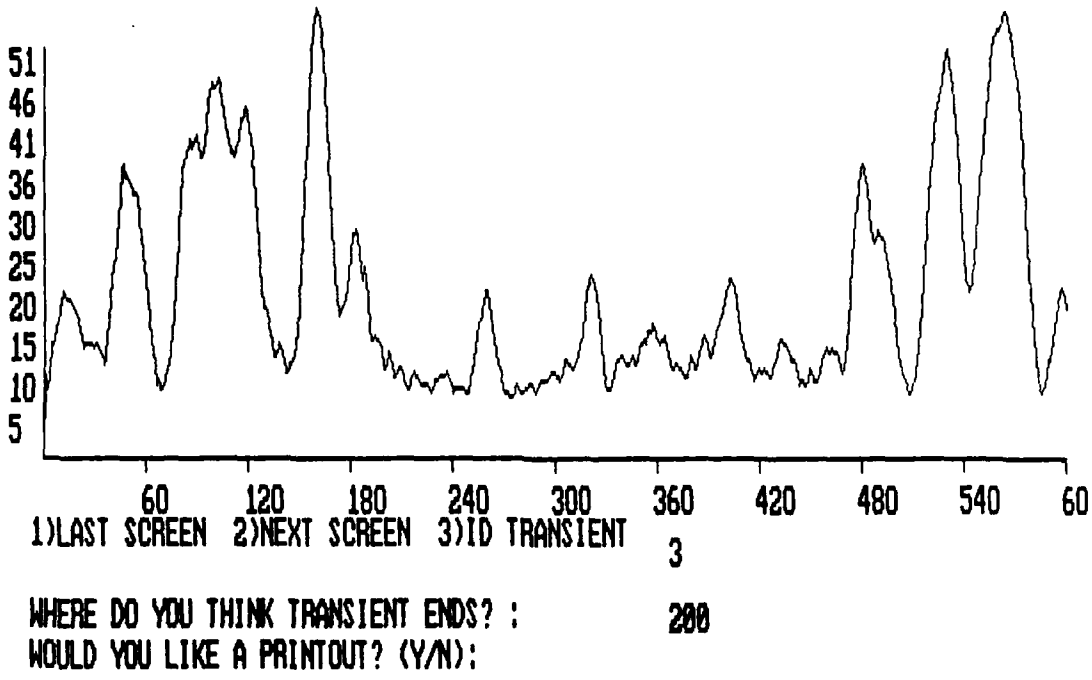

Figure 6. Output Analysis Program Graph of Data

The user is given the option to perform batch means for a single file, independent replications for multiple files, or return to the original data menu. The batch means procedure uses Fishman's technique (8:514), while the independent replications procedure follows Welch's equations (27:294-296), both of which are discussed in Chapter II. The program then provides the mean, its associated variance, and confidence interval data as shown in Figures 7 and 8.

The major advantage of this program is that it provides the user the ability to assess multiple data files, and instantly see the impact of selecting different cutoff values. This makes sensitivity analysis very feasible, with very little additional effort required.

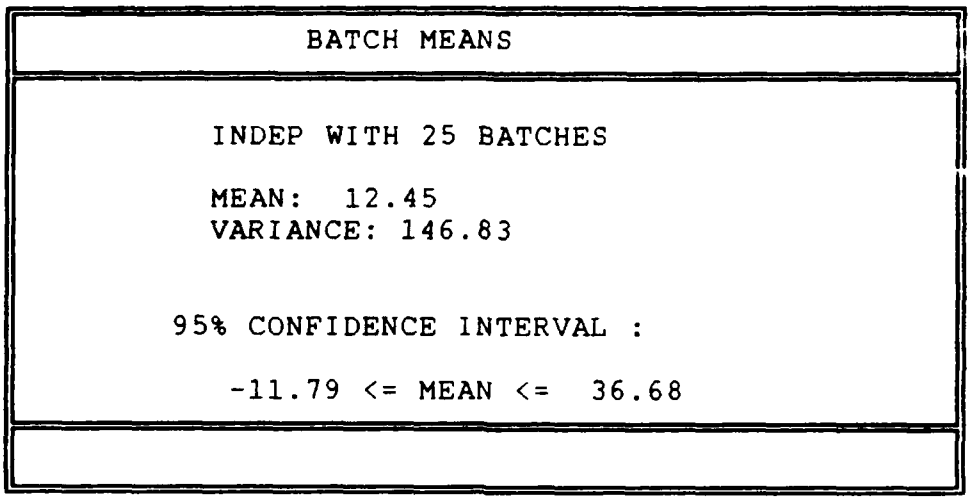

Figure 7. Batch Means Output Analysis Data

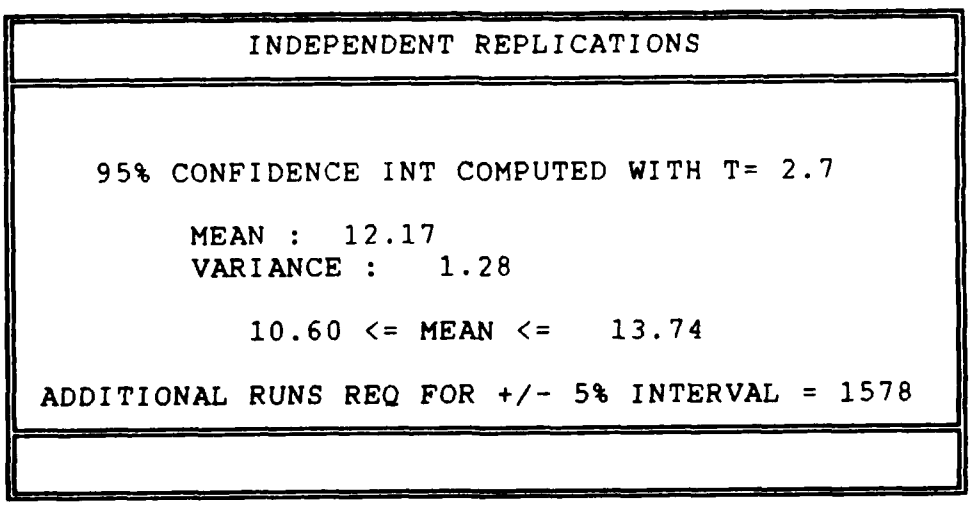

Figure 8. Independent Replications Output Analysis Data

Application of Kalman filtering

The key to implementing Kalman filtering is to specify the characteristics of the system which is to be modelled correctly. To accomplish this, the software package FORECAST MASTER (9:7-1 - 7-13), and in particular, the state space modelling option, was employed. FORECAST MASTER uses the Goodrich-Larimore algorithm (9:7-8) to fit the appropriate ARMA model to the observed data.

This ARMA model was then converted into the appropriate state space representation for Kalman filtering (9:7-8 - 7-11; 11:101; 25:92). The user's manual addresses the differences in their algorithm from the Akaike-Mehra algorithm, referencing Mehra's work on the single adaptive filter (18:175). The program provides the state transition matrix G_t, the Kalman gain matrix $[G_tK]$, and the measurement (forecast) matrix F_t of Eqs (36) and (37), referenced in Chapter II.

In evaluating the Kalman filter performance, only the first 1000 data values from each of the 50 files (see Figure 4) were used. These are exactly the same data values used to estimate the mean with the batch means technique.

The one stipulation to use the adaptive Kalman filter was that the data had to be equally spaced in time. It was, therefore, interpolated. The two techniques attempted were a cubic spline and linear interpolation between successive

points. Both of these techniques are addressed in detail in the data formatting section below.

Once the equally spaced data was obtained, the appropriate state space model, and Kalman filter matrices had to be computed. Because the filter's matrices will become constant when the system has reached steady state, observing the changes within the Kalman gain matrix provides information on when steady state has been achieved. At this point, FORECAST MASTER's data limitation of 2000 observations was encountered.

In order to evaluate the filter, data sets of 2000 observations were generated in several ways. The first data set formed, consisted of the last 200 observations from the first ten output files. The second set contained the last 100 observations of the first 20 files. The third set was comprised of the last 50 observations of 40 files, while the fourth set contained the last 40 observations of all 50 files. The computer program used to obtain these values is shown in Appendix D. Each of these four data sets was then evaluated by FORECAST MASTER, for autocorrelation and to determine the G_t, $[G,K]$, and F_t matrices.

In deciding the appropriate order for the ARMA model, and therefore, the Kalman filter, the data's autocorrelation was assessed. Figure 9, suggests a second order Markov process (1:206; 16:183), which indicates that the

appropriate state space model for the data should have at least two states. Evaluation of the Akaike Identification Criteria (9:6-4) indicated that the second order model provided the best "fit" of the data.

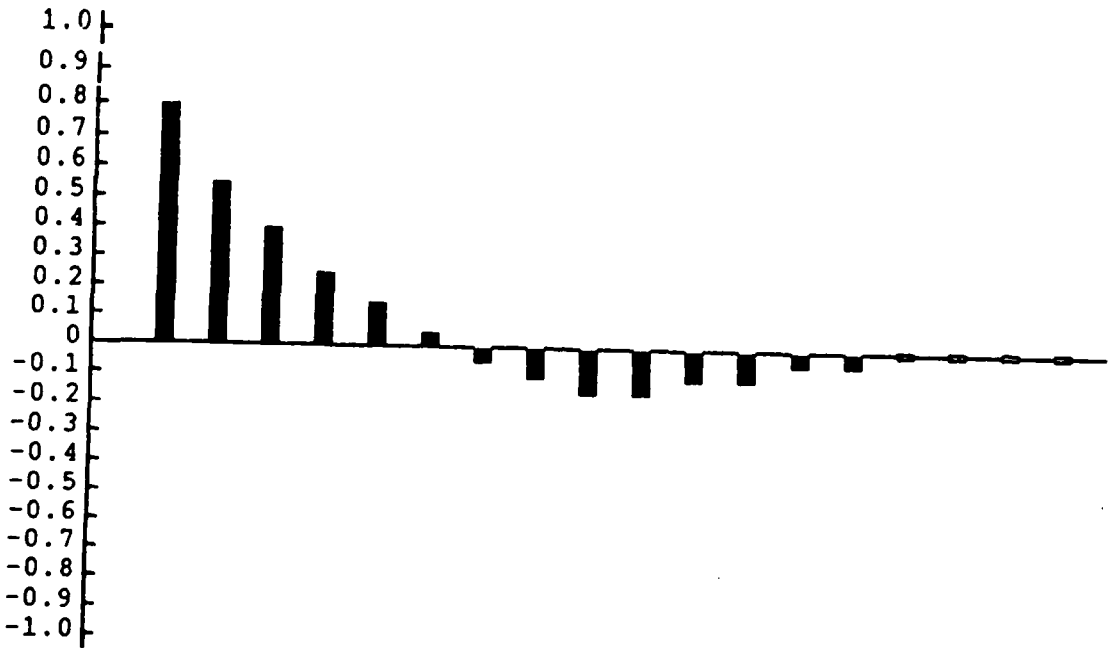

Figure 9. Autocorrelation Plot of Equally Spaced Output, I=10

In examining the Kalman filter matrices generated, the values for the last two sets of data were effectively the same. This indicated that the best possible model for the data had been achieved. The resultant Kalman filter matrices, obtained from FORECAST MASTER, are shown in Figure 10, in a sample output format from the program, where the program's notation differs from Meinhold and Singpurwalla

(19:123) referenced in Chapter II. In the program output the F matrix represents the transition matrix (G_t), the K matrix remains the Kalman gain matrix K (G_tK], and the H matrix represents the measurement (forecast) matrix (F_t).

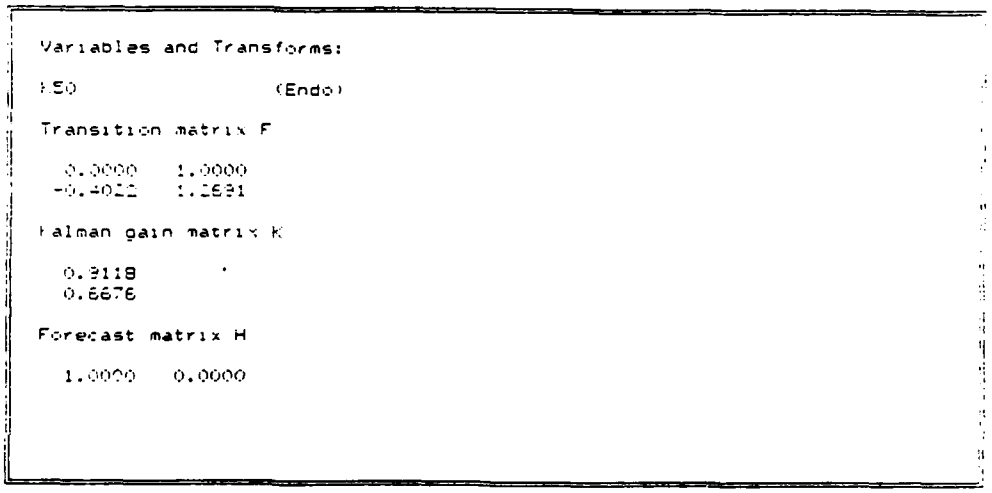

Figure 10. Kalman Filter Values from FORECAST MASTER

The state space modelling option also provided a screen with statistical information on the data set used for the model (9:7-4), shown in Figure 11. The information provided gives insight into the basic characteristics of the data which was modelled. The fist two lines indicate the average autocorrelation between each of the residual values and residuals which occurred a "lag" of 1 to 12 before. The

AUTOCORRELATIONS OF LAGGED RESIDUAL ERRORS

Lag:	1	2	3	4	5	6	7	8	9	10	11	12
K50	.00	.01	-.00	.03	-.00	.02	-.09	-.05	-.08	-.04	-.04	-.00
	-.00	.01	.02	-.01	-.02	.01						

Statistic	Value	Probability
Number of observations	1998	
Mean value of K50	11.162600	
Standard deviation of K50	10.331002	
Standard error of forecast	6.435969	
R-square (corrected for mean)	0.612484	
F(4,1994)	787.897226	1.000000
Adjusted R-square	0.611706	
Ljung-Box Test = Chisq(14)	45.811991	0.999970
Durbin-Watson Statistic	1.996797	0.056935
Standardized AIC Statistic	0.623600	
Standardized BIC Statistic	0.627105	

Figure 11. State Space Modelling Statistics

best way to describe the concept of lag is with an example. Residual 4 is "lag 1" from residual 5, while it is also "lag 3" from residual 7. The intent of reviewing these autocorrelations is to ensure that after the data has been modelled, that the residuals do reflect a "white noise" process with little to no correlation. The table gives the number of observations used was 1998, of 2000 available. This was due to the fact that the second order model required two previous values to make the next prediction, so the first two data values could not have predictions. The next two entries provide the mean of the data values provided and the standard deviation of that mean. The standard error of the forecast is particularly important, representing the square root of the underlying system variance (R_t) shown in Equation (37) of Chapter II.

The R-square, F value, and adjusted R-square all apply to the adequacy of the Kalman filter in modelling the data, where an R-square value of 1.0 indicates a perfect fit (21:423). Both the Ljung-Box test and the Durbin-Watson statistic indicate the "degree to which the error residuals over the historical sample adhere to the hypothesis that they are independently normally distributed, with zero mean" (9:6-5). The Durbin-Watson test statistic is "the ratio of the sum of squares of the first differences of the errors, divided by the sum of squares of the errors" (9:6-5). As an indication, the hypothesis is supported more as this value approaches 2.0. The Ljung-Box test is similar but reviews several lag's autocorrelation, rather than just the first (9:6-6).

Data Formatting

The batch means and independent replications techniques required only the entities' actual time in system, while Kalman filtering required these values along with the time between observations. This resulted in the output files, generated by the SLAM II program, being operated upon at least twice to get the necessary formats. The data files generated, and the programs used, are shown in Figure 12, and described below.

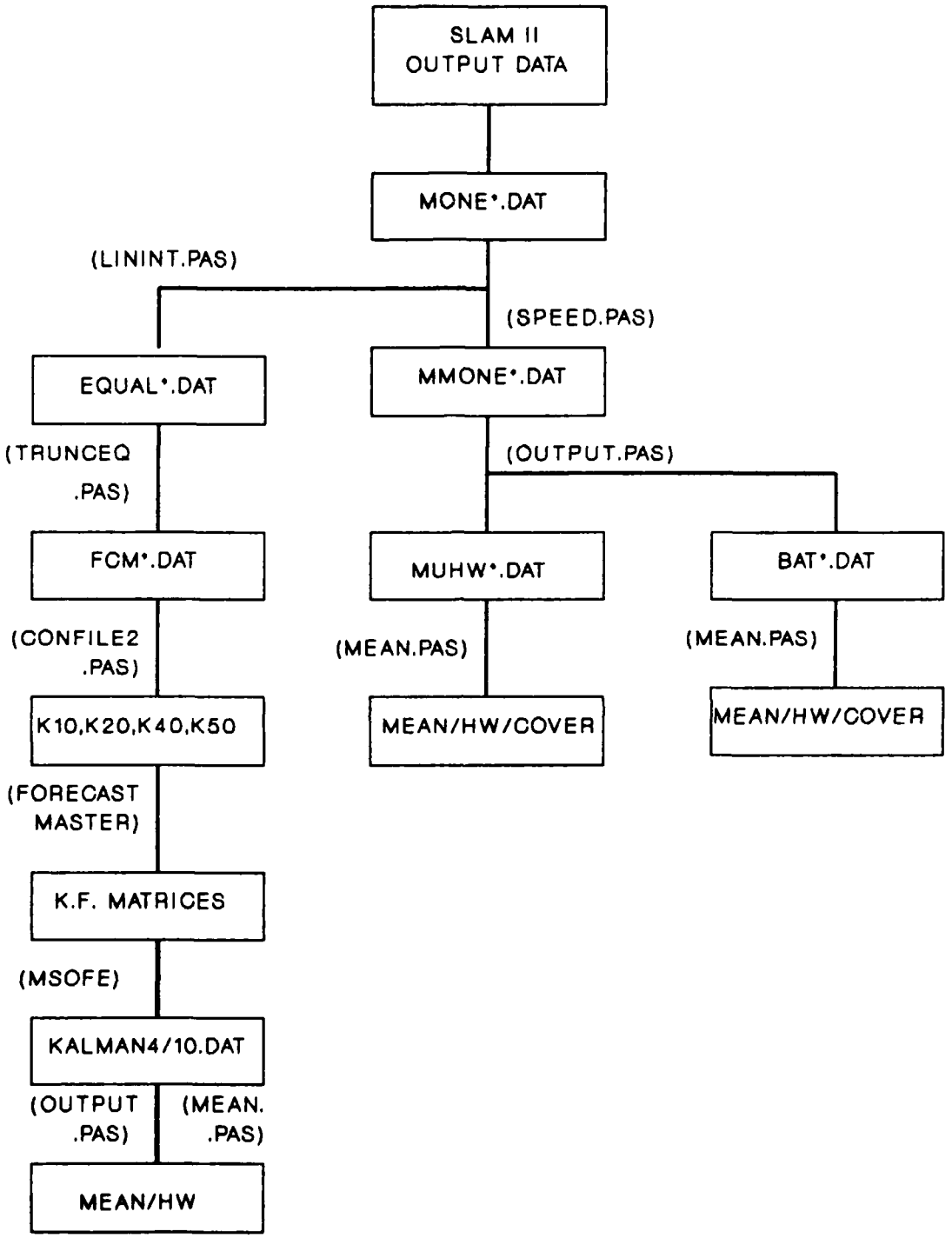

Figure 12. Flowchart of Data Formatting

The entries within each of the blocks on the flowchart indicate the file names given to the data files generated. The entries within the parentheses () indicate the programs used to develop each data file. Table 1 lists all of the programs above and the purpose of their output files.

PROGRAM	PURPOSE
LININT.PAS	Generates a file of linearly interpolated data at a specified sample rate from an existing data file.
SPEED.PAS	Produces a data file from the original SLAM II output which contains only the time in system values.
TRUNCEQ.PAS	Truncates the transient portion of the data from the data file.
CONFILE2.PAS	Concatenates portions of the steady state data files to form a file of 2000 values to be evaluated by FORECAST MASTER.
FORECAST MASTER ™	State space modelling option was incorporated to generate the appropriate Kalman filter matrices for the files generated with CONFILE2.PAS (9:1).
MSOFE	Generates the Kalman filter next step predictions, based on the matrices from FORECAST MASTER and the equally spaced data from LININT.PAS (4:1).
OUTPUT.PAS	The Output Analysis Program generates the mean and its associated variance of the original and Kalman filter predicted time in system values.
MEAN.PAS	Calculates the grand mean and its associated variance for the means of the Kalman filter predictions, with cutoff values from 0 to 25% at 5% increments.

Table 1. Programs Used and Their Purposes

Evaluating the data files for the Kalman filtering approach demonstrated that the cubic spline interpolation, from the software package MATRIX$_x$ (12:4-34), was not feasible due to the data's high variability. The cubic spline criteria to pass through every data point with a continuous function (3:118) caused the interpolation to greatly exaggerate the data. Where the original data varied from approximately 1/1000 to 64 minutes, the interpolated data ranged from -2700 to 6700 minutes. At this point, the linear interpolation method was employed. Its values were restricted to the range of the actual data, as the interpolated values were drawn from a line connecting original, sequential data points. A plot of an original data set and a partial plot of the fitted cubic spline are shown in Figures 13 and 14.

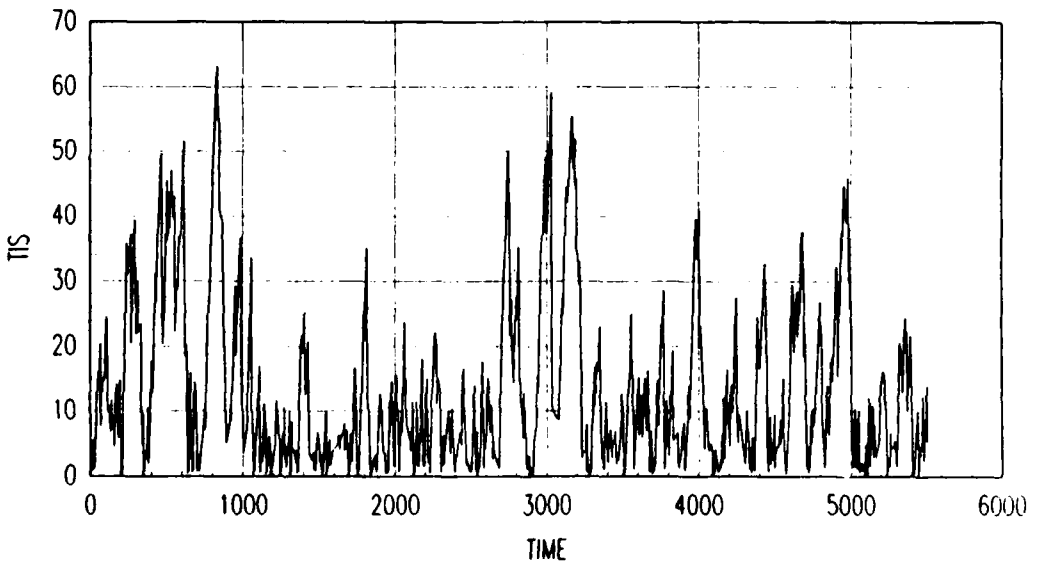

Figure 13. Plot of Initial Output Data

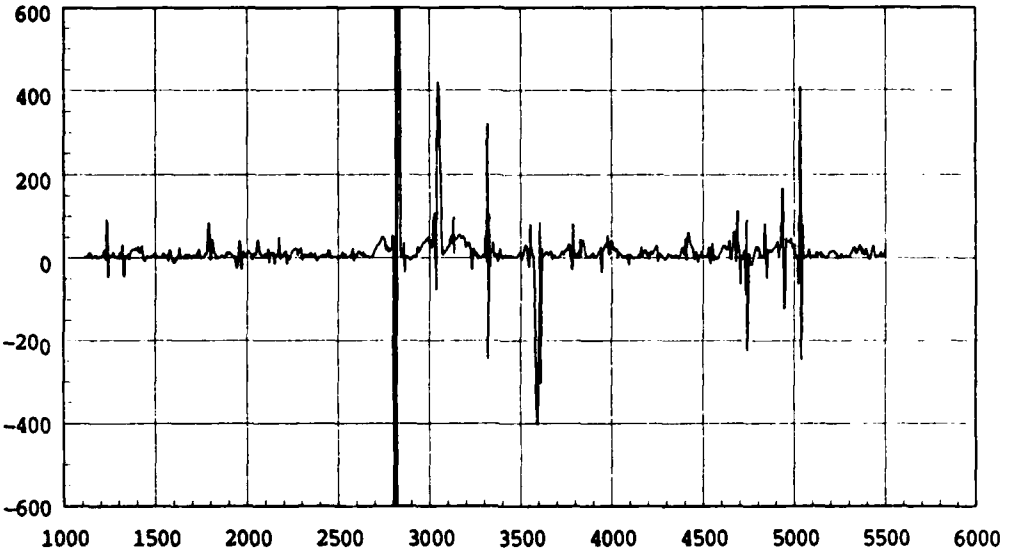

Figure 14. Partial Cubic Spline Interpolation of Output

The linear interpolation program (LININT.PAS) provided interpolated values at each tenth time unit. This program built the files to be evaluated, in turn, by the FORECAST MASTER program. Figure 15 is a plot of the linearly interpolated data from the data shown in Figure 14.

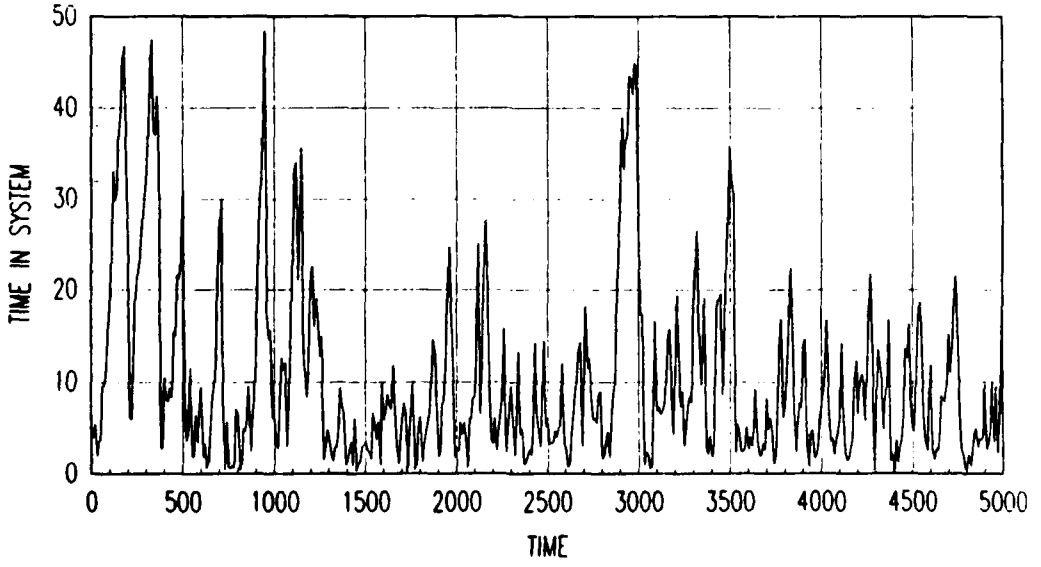

Figure 15. Plot of Linearly Interpolated Data

When the mean values for the equally spaced data were compared to the batch means for the corresponding original output data, they were consistently lower. This is due to the dynamics of the queueing system. The requirement for equally spaced data automatically biases the linearly interpolated values low. When the queue is full, leading to higher times in system, then the entities exit at an exponential rate of 15/hr, the service rate. When the queue is empty, the time between observations is higher, while the total time in system for the entity decreases. This resulted in the lower time in system entities having a larger distance, on a time axis, between them. This, in turn, meant that as the time between samples decreased, the percentage of time that these lower time in system values were measured increased. As is shown in Chapter IV, this caused the mean to be biased lower as the interpolated sampling rate increased.

Summary

Now that the data formats and techniques have been discussed, the three techniques must be compared. Chapter IV presents the results of each technique. Comparisons were accomplished for cutoff values of 0 to 25% of the data, based on the calculation of the system's steady state mean, and its 95% confidence interval coverage rate of the analytic and system average means. Chapter V presents

conclusions and recommendations, along with identifying some of the problems which needed to be resolved in the implementation of the Kalman filter technique.

IV. Results and Discussions

Output Analysis

This chapter presents the results of the batch means, independent replications, and Kalman filtering techniques in assessing the system's steady state mean and variance. Comparisons are based upon computation of the grand mean, the average size of the predicted 95% confidence interval halfwidth, and the percentage of time the confidence intervals for each run covered the grand and analytic means for each cutoff value. The grand mean represents the average of the means from each of the 50 data files, where the analytic mean is the steady state mean computed below.

To predict the system's steady state mean analytically, equations from Introduction to Probability Models, by Ross are used (24:306-314). The statistic of interest was the time spent in the system by each individual entity, represented by the letter W. To determine this value, required the two variables which represent the system's average arrival rate, λ, and average service rate, μ.

The arrival rate represents the average number of entities which enter the system during a specific time interval. The time between arrivals for this research was drawn from an exponential distribution with a mean of 6, resulting in an average arrival rate of 10 entities/hour.

The service rate indicates the average amount of time that the entity must spend within a particular service activity. This value was drawn from an exponential distribution with a mean value of 4, making the average service rate 15 entities/hour.

Given these definitions and the equations from Ross (24:306-314), the average time each entity spends in the system can be determined with the following equations.

Arrival Rate $\lambda = 10/hr$

Service Rate $\mu = 15/hr$

$$W = 1/(\mu - \lambda) \tag{47}$$

$$W = 1/(5/hr) = .2\ hr = 12\ minutes$$

The analytic mean of 12 minutes was then used as a benchmark for the output analysis techniques.

Batch Means. The simulation output files were sorted to obtain only the first 1000 actual time in system (TIS) observations, for each of the 50 data files. The SLAM II output data format is shown again in Figure 16.

This data was loaded into the Output Analysis Program, which graphed the moving average with a window width of 10, as shown in Figure 17. This graph was then evaluated to determine the extent of the transient.

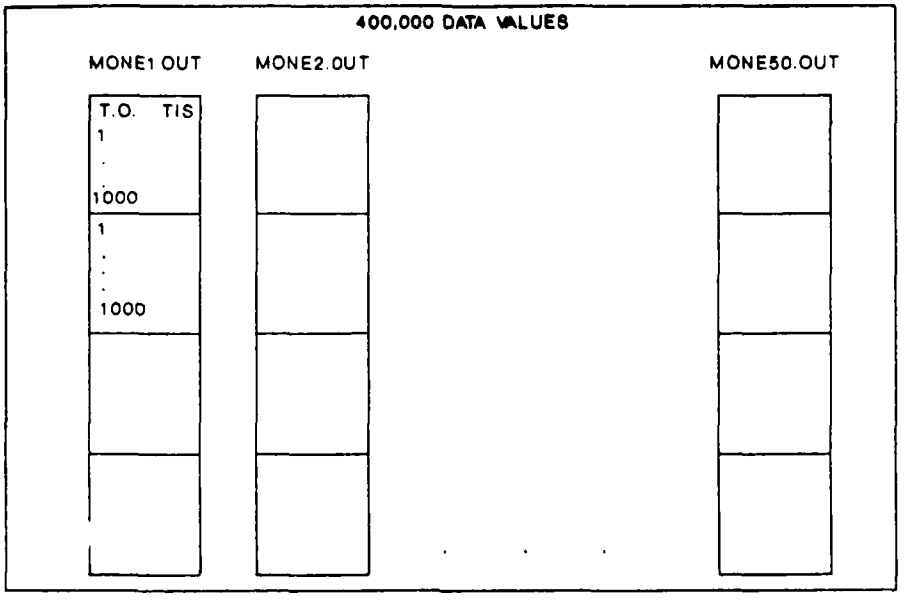

Figure 16. Configuration of SLAM II Output

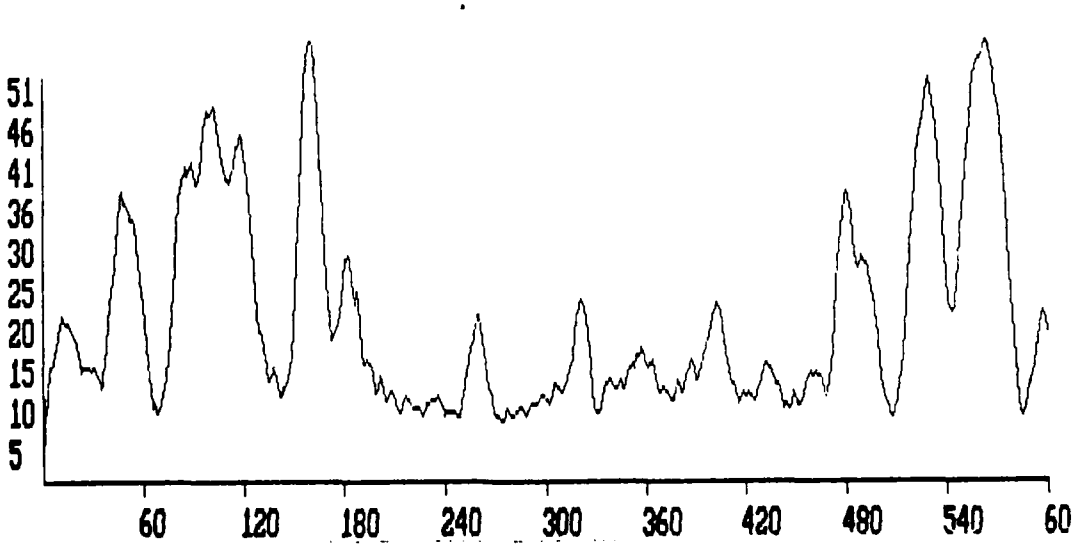

Figure 17. Graph of Moving Averaged Output Data

Upon evaluating the above figure, it is unclear as to where exactly the transient period ends. Because of this, various cutoff values were tried. The batch means algorithm, developed by Welch (27:307) and described in Chapter II, was used for each cutoff value to obtain the values listed in Table 2.

The grand mean and average halfwidth represent the average of the 50-run's mean and halfwidth values for each cutoff. Cutoff values on this table are by number of observations, while coverage rates indicate the percentage of the 50 confidence intervals for each cutoff which covered the corresponding grand mean and analytic mean of 12.

CUTOFF	GRAND MEAN(S.D.)	AVG HALFWIDTH	(MEAN) % COVER	(12) % COVER	AVG BATCH SIZE
0	12.63(1.3)	2.47	88	98	28.67
50	12.55(1.4)	3.05	90	96	28.48
100	12.10(1.3)	2.91	94	94	28.80
150	12.18(1.4)	3.23	94	96	29.12
200	11.21(1.1)	2.55	100	84	22.08
250	11.35(1.1)	2.59	100	92	21.44

Table 2. Batch Means Performance Results

Reviewing the values contained in Table 2, several interesting facts come to light. First, the coverage of the grand mean increases as more data is truncated, while the coverage of the analytic mean decreases. This indicates that the process is still operating somewhere in the transient phase of the simulation, with a bias high at the beginning. Second, despite the loss of data, the last two halfwidth values are the smallest. This can be attributed to two reasons: decreasing variance as the system warms up, and the reduced number of observations per batch allowing for more batches, and thus a tighter interval.

Independent replications. The independent replications technique requires two or more runs of simulation output. In this research, the technique was performed with four independent output files of 1000 data values concatenated together (see Figure 16).

The Output Analysis Program evaluated these files using the equations developed by Welch (27:282) as presented in Chapter II. As in the case of batch means, the extent of the transient period was not easily distinguished. This can be seen in Figure 18. Again, multiple cutoff values, ranging from 0 to 250, were tried. Overall average values are shown, for each cutoff in Table 3.

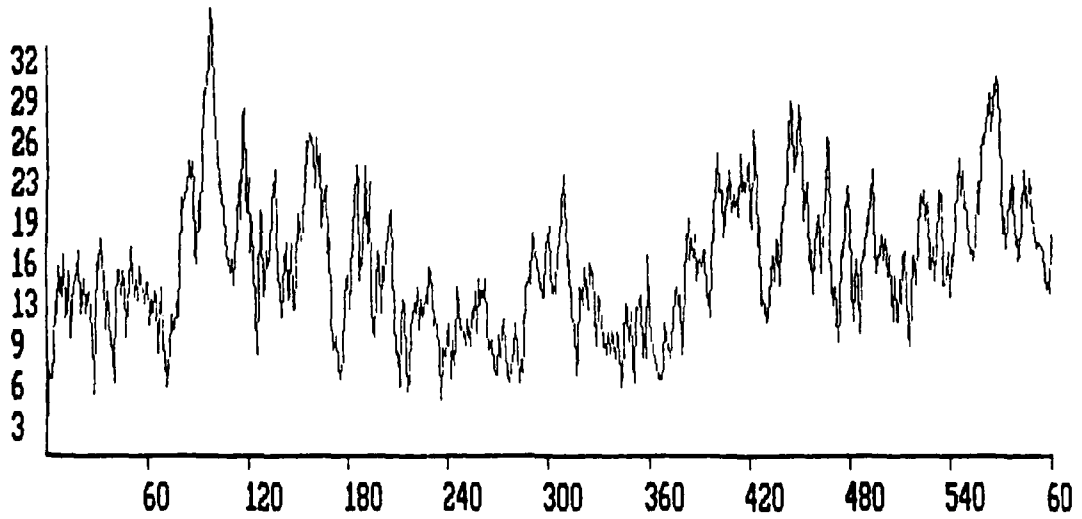

Figure 18. Graph of Averaged Replications Data

CUTOFF	GRAND MEAN(S.D.)	AVG HALFWIDTH	(MEAN) % COVER	(12) % COVER
0	12.55(.93)	1.29	68	62
50	12.41(1.1)	2.47	72	76
100	12.28(.96)	1.68	84	86
150	12.08(.97)	1.50	80	78
200	11.46(.82)	1.35	78	66
250	11.16(.91)	1.15	70	56

Table 3. Independent Replications Results

The results shown in Table 3 demonstrate the problems which can be encountered if too much data is used when the system is still in a transient state. The large amount of data caused the confidence intervals to be drawn in too tightly about the estimated means. Because the system was still in the transient, these reduced confidence intervals were centered at values below the true mean. This resulted in the coverage rates, of both the grand and analytic means, to be much lower than those for batch means, indicating that data would have to be obtained further out in the run for independent replications to be used effectively. As a note of interest, if the true mean had been unknown, the interval would have been thought to have contained the true mean 95% of the time, based on the calculations presented in Equation (10), and the average time in system would have been biased low.

Figure 19 presents a graphical depiction of the mean values obtained by both batch means and independent replications, versus the analytic steady state mean of 12. This figure indicates that the value of the estimated system mean drops as the cutoff value is increased. This was directly the opposite of the result expected, giving two indications. First, the system was biased high early in the transient, due to a large initial buildup in the queue at the start of each simulation. Second, the values with

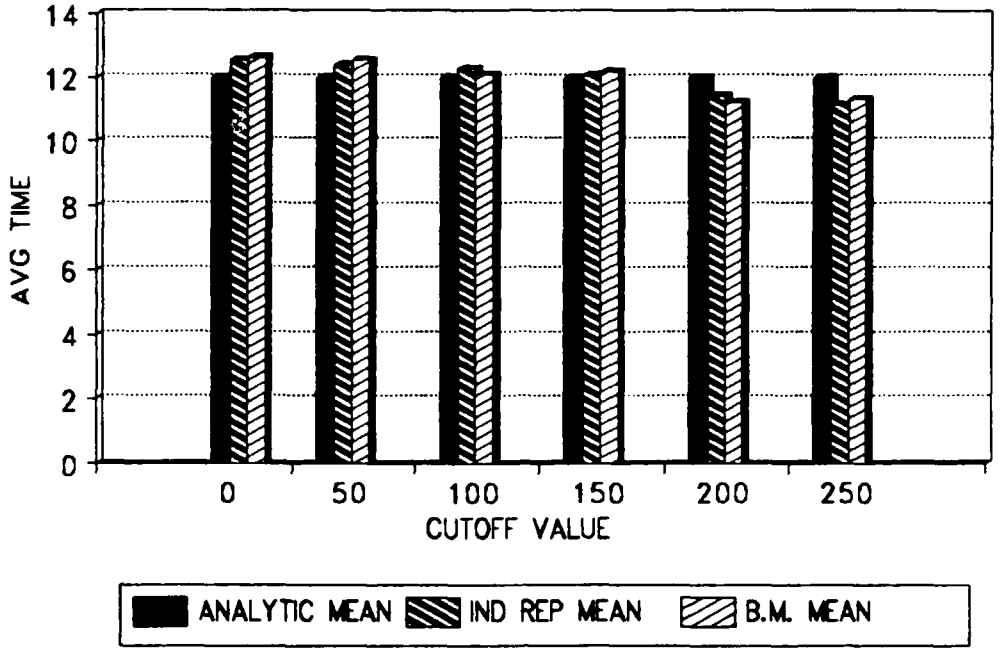

Figure 19. Comparison of Estimated and Analytic Means

higher cutoffs had mean values lower than the analytic mean, indicating the process was probably still operating in the transient. These two facts must be kept in mind while evaluating the three techniques.

Kalman Filter. The original SLAM II output data was linearly interpolated at intervals of 10 (I=10) and then evaluated using FORECAST MASTER (9:7-8) to obtain the Kalman filter data shown in Figure 20.

These matrices and the data were then run through the computer program Multimode Simulation for Optimal Filter Evaluation (MSOFE) (4:1). Forecasts generated were evaluated by the batch means portion of the output analysis

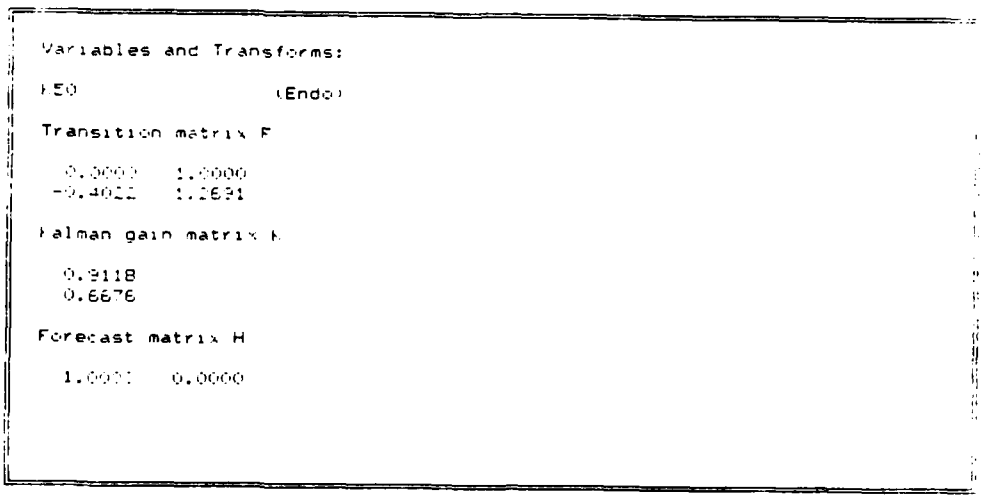

Figure 20. FORECAST MASTER Output on the Kalman Filter, I=10

program to obtain mean and 95% confidence level halfwidth information for each cutoff value. Cutoff intervals were scaled 5% of the observations, and performed from 0 to 25% of the data to evaluate their effect. For a data set of 1000, this would be 50, for 500 it is 25, and, as will be used later, for 1300 it is 65. Table 4 displays the grand mean (standard deviation) and average 95% halfwidth values, by cutoff, for data interpolated at each tenth minute. To calculate the mean and halfwidth values for the Kalman filter predictions, the batch means portion of the Output Analysis Program was employed. The coverage values reflect

the percent of the confidence intervals which cover the grand mean and the analytic mean of 12.

CUTOFF	GRAND MEAN(S.D.)	AVG HALFWIDTH	(MEAN) % COVER	(12) % COVER	# PER BATCH
0	11.23(1.2)	2.74	88	84	15.4
25	11.18(1.3)	2.96	92	80	16.5
50	10.87(1.2)	2.83	90	78	15.0
75	10.55(1.2)	2.80	90	69	13.9
100	10.42(1.0)	2.87	90	76	15.0
125	9.63(.99)	2.41	100	52	11.8

Table 4. Kalman Filter Performance Results

Again, as in batch means, the number of observations required per batch drops as the cutoff value increases. This explains the tightest confidence interval corresponding to the largest cutoff, even though less total data was used. The coverage rates of the grand mean were comparable to those for the batch means results on the original data. The coverage rates for the analytic mean, however, were much lower. This was because the value for the Kalman filter estimate of the mean was lower than those for either batch means or independent replications.

Review of the equally spaced data showed that after linear interpolation the overall data mean was lower than originally. The reason appeared to be that the sampling

rate, 6/hr, was so much smaller than the service rate of 15/hr. Normally, this would imply that the sampling rate should be increased. This, however, assumes that samples are taken directly from the process, and not interpolated from existing data. The effect of increasing the sampling rate was then investigated.

The SLAM II output was once more interpolated; this time with a time between observations of 4 minutes. The interpolated values were then evaluated by FORECAST MASTER to obtain the Kalman filter data shown in Figure 21.

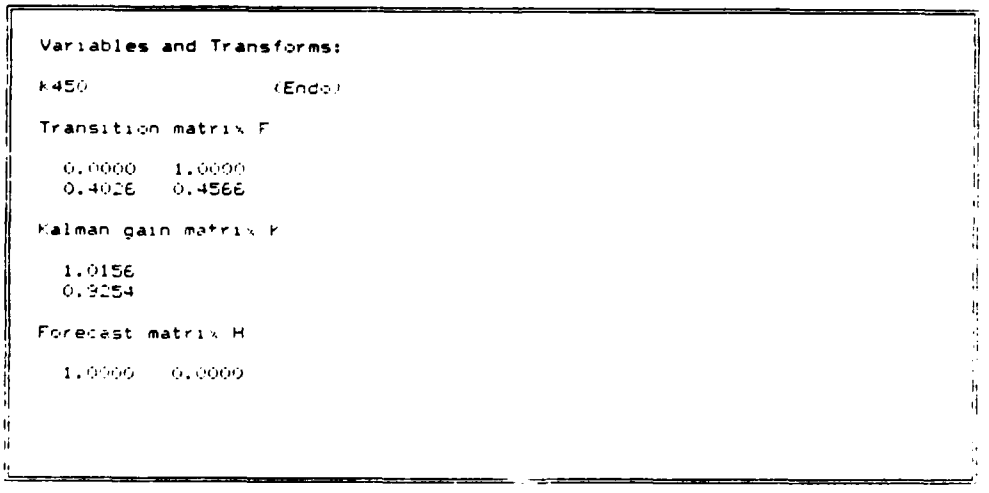

Figure 21. FORECAST MASTER Output on Kalman filter, I=4

CUTOFF	GRAND MEAN(S.D.)	AVG HALFWIDTH	(MEAN) % COVER	(12) % COVER	# PER BATCH
0	9.26(.87)	3.83	97	73	72.7
65	9.31(1.1)	4.15	97	73	85.6
130	8.51(.99)	3.82	100	58	75.4
195	8.45(1.0)	4.22	100	55	86.6
260	7.71(.98)	3.91	100	41	78.7
325	6.76(.80)	3.08	100	10	50.1

Table 5. Kalman Filter Results, I=4

Evaluating in the same manner as for the interval of 10 resulted in the data shown in Table 5, which shows the lower means and coverage rates as a result of increasing the sampling rate, using linear interpolation. The two major impacts come from the reduction of the mean and the size of the batches required to pass the independence test.

The decrease in the mean value is due to the equal weighting of the interpolated values. The problem arose because entities which had to wait in the queue exited the system approximately every 4 minutes, matching the service rate exactly. Those entities that didn't have to wait exited, on the average, with a higher time between observations and lower times in system. Because the time

between observations for the lower "time in system" values was higher, the equally spaced observations landed on the lines connecting these lower points more often. Histograms are shown, for one of the data sets, in all three formats: the original SLAM II output, linearly interpolated for I=10, and linearly interpolated for I=4. These are shown in Figures 22, 23, and 24 respectively, where the time axis remains the same, but the frequency scale changes.

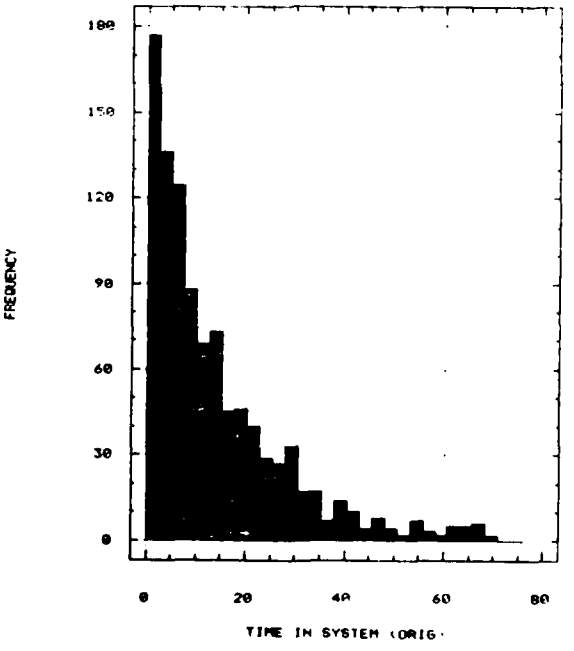

Figure 22. Histogram of Original Output Distribution

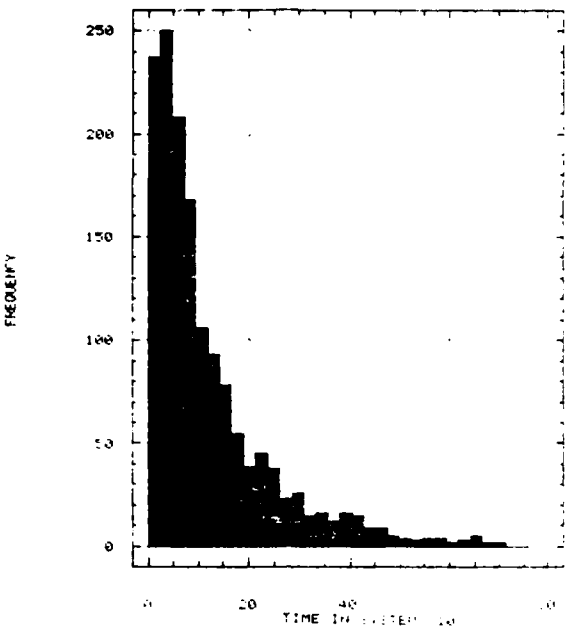

Figure 23. Histogram of I=10 Interpolated Data Distribution

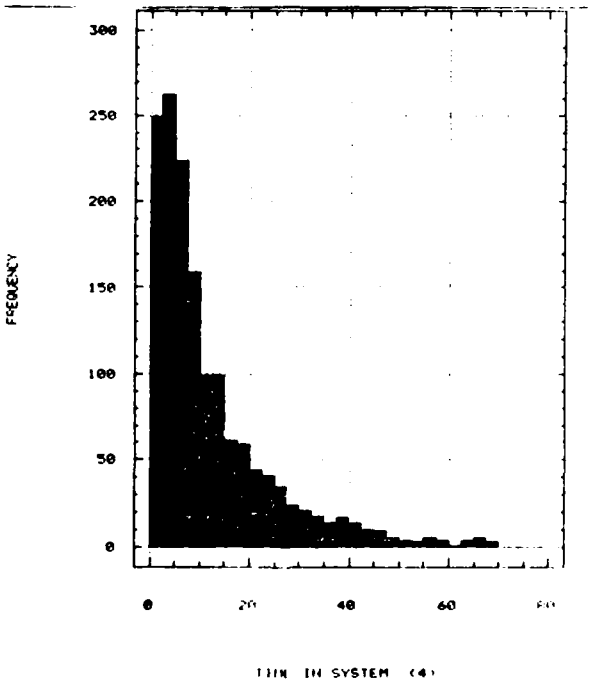

Figure 24. Histogram of I=4 Interpolated Data Distribution

The histograms reflect the increased weighting of the lower time in system observations with increased sampling. Note that the frequency value of 150 on the original data histogram represents 15% of the entire population, where for the I=10 data it is 30%, and for the I=4 data it is approximately 11.5%. This does not affect the interpretation, however, of how the distribution is moving to the left, indicating that the average of the observations decreased as the sampling rate increased. Appendix E contains a two dimensional array with the time between observations vs. time in system.

Now that the decreasing mean has been explained, the large difference in the batches required must be addressed. The answer again lies in the time between observations. When the interval was 10, the data was autocorrelated as a typical second-order Markov process. (16:183; 1:206) This autocorrelation is shown, in a plot from FORECAST MASTER, in Figure 25.

When the time between interpolated points is reduced to four, the bias of the higher time between observations values comes into play once more. The lower time interval causes the interpolated data to be highly autocorrelated, masking much of the underlying system's mechanics. This autocorrelation is shown in Figure 26.

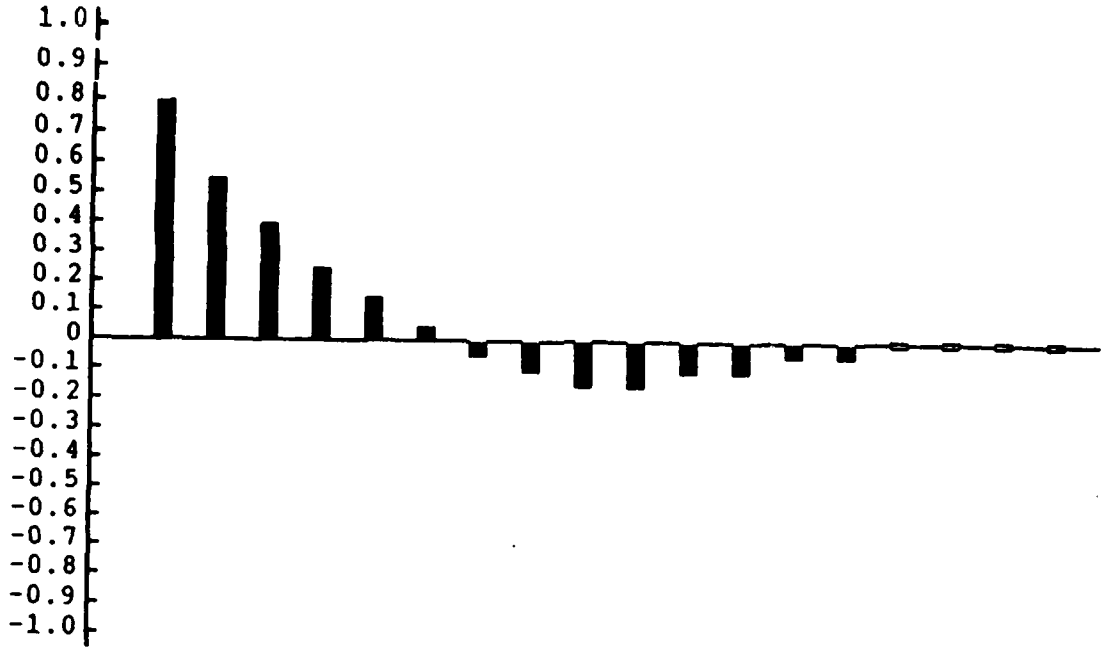

Figure 25. Autocorrelation of I=10 Data

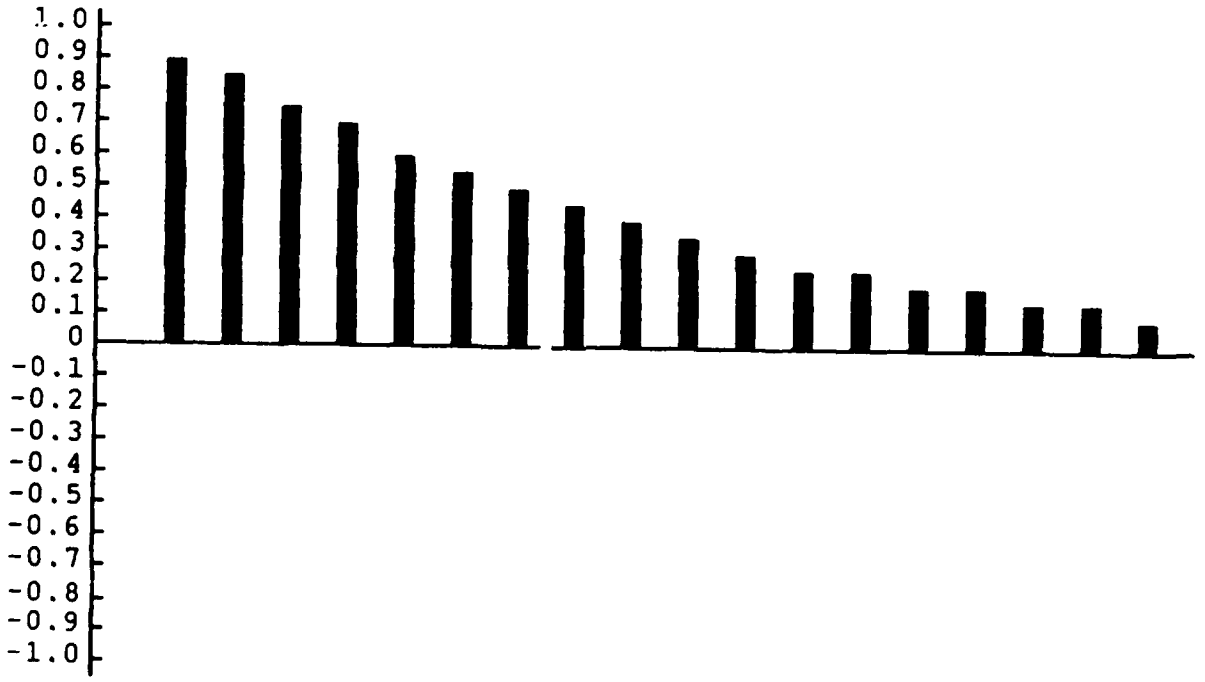

Figure 26. Autocorrelation of I=4 Data

Although no units are displayed, the autocorrelations in Figure 25 are spaced at time intervals of 10 minutes, while those in Figure 26 are every 4 minutes. A direct comparison of the lag autocorrelations is, therefore, impossible.

With the sources of the bias identified, the means and average delta between the original and interpolated means were computed. These are shown in Tables 6 and 7, where the cutoff values represent the percentage of the total observations included in the cutoff. The number 0, therefore, reflects the cutoff value 0 for all methods while 5(%) indicates 50, 25, or 65, depending on the method.

	CUTOFF Percentage					
I	0	5	10	15	20	25
ORIG	12.63	12.55	12.10	12.18	11.21	11.35
10	11.23	11.18	10.87	10.55	10.42	9.63
4	9.26	9.31	8.51	8.45	7.71	6.76

Table 6. Comparison of Mean Values

I	0	5	10	15	20	25	AVG
10	1.40	1.32	1.23	1.63	0.79	1.72	1.35
4	3.37	3.24	3.59	3.73	3.50	4.59	3.67

Table 7. Differences between Mean Values

The low coverage rates shown in Tables 4 and 5 are influenced directly by the interpolation underestimating the mean. The effect on the coverage rate of adding the average bias, δ = 1.35 for I=10 and δ = 3.67 for I=4, to the estimated mean values is shown in Table 8.

Coverage Rates

CUTOFF	(4) MEAN	(4) MEAN + δ	(10) MEAN	(10) MEAN + δ
0	73	100	84	94
5	73	100	80	94
10	58	100	78	94
15	55	100	69	90
20	41	100	76	90
25	10	87	52	82

Table 8. Coverage Rates with Delta Added In

The dramatic increase in percent coverage of the analytic mean value of 12 indicates that the filter is performing well, but is biased by the process by which the data was obtained.

Calculation of System Variance. Currently, insight to the underlying system variance is obtained through observing the variance of the estimated mean and through the use of pseudovalues, in a technique known as Jackknifing (15:227; 22:737). The Jackknifing technique effectively computes

pseudovalues which represent calculations of the variance as each one of the data values is individually removed from the data file. The intent is to remove some of the sensitivity of the standard variance calculations to departures from normality, giving a better indication of the true system performance. The Kalman filtering output from FORECAST MASTER, however, allows for easy computation of the system variance.

The computation is possible due to the assumptions that the system has attained steady state, and that the measurement values (time in system) are perfect, in that they are explicitly known. Given these facts, the standard error of the forecasts was taken from the output data shown in Figure 20. This value represents the best estimate of the system variance before the next observation is seen, R_t. It will be used along with the matrices from Figure 10 to compute the true system variance at the time of measurement, Σ_t using Equation (42).

$$\Sigma = (R_t) - K*F_t*(R_t) \qquad (42)$$

where K has been substituted for $R_tF_t'(V_t + F_tR_tF_t')^{-1}$, the Kalman gain matrix. Because of the nature of the F_t matrix, [1,0], the value of Σ_{11}, the variance associated with the next prediction, can be computed using Equation (43).

$$\Sigma_{11} = (R_{11}) - K_1*(1)*(R_{11}) \qquad (43)$$

$$\Sigma_{11} = 3.66$$

This value represents the underlying true variance of the system just after an observation is made, implying a standard deviation of 1.91. This can not be compared to the average standard deviations of the mean computed by batch means on the Kalman filtered data. It represents, instead, the true underlying variance of the system. This variance fluctuates in a sawtooth pattern as shown in Figure 27 (16:223).

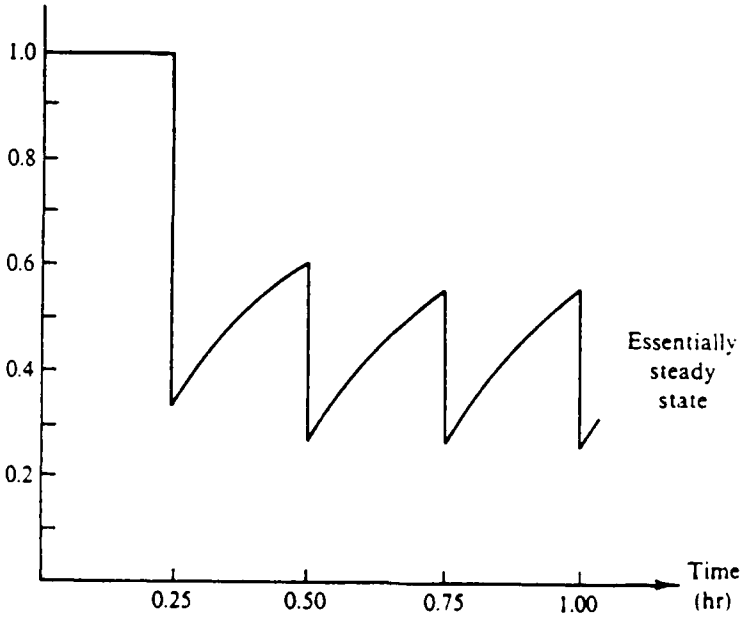

Figure 27. Error Variance Time History

The variance in Figure 27 begins at a specified level, dropping straight down as the first observation is taken. After the observation the variance increases asymptotically toward the initial variance until the next observation is taken. This process continues until some steady state is achieved, where the top, R_{ss}, and the bottom, Σ_{ss}, of the sawtooth become constant. These values now represent the range over which the variance of the system can travel. These values are useful for comparing the variance of one system to another to assess the variability of the process.

In summary, each of the techniques provided some insight into the M/M/1 queue which was modelled. The independent replications' results were biased low by the fact that the process was still within the transient, while the Kalman filter experienced problems with the way in which the data was interpolated. Recommendations to correct the interpolation errors are addressed in Chapter V.

V. Conclusions and Recommendations

Conclusions

The objectives set forth for this thesis were the development of a computer program to automate the batch means and independent replications simulation output analysis techniques and the evaluation of possible benefits of Kalman filtering in the same area. Both of these goals were achieved, while several areas for future research were identified.

The Output Analysis Program provides the analyst with the ability to assess the system's steady state mean and its associated variance from the simulation output. Designed for implementation on Air Force standard micro-computers, it allows the analyst to perform the following functions from within one program.

1. Data averaging across replications
2. Moving averages within a single long run
3. Transient identification, from a graph of the data
4. Data truncation of the transient
5. Generation of a file for steady state data
6. Batch means evaluation of a single run
7. Independent replications evaluation for multiple runs

With these capabilities, the analyst can also instantly assess the impact of different cutoff values.

The data used to evaluate the simulation output included the system's warm-up from its initial state of empty and idle. This was done purposely to reflect real world conditions and assess the effects of the transient. This approach also reduced the length of the SLAM II runs required, keeping system costs lower.

In evaluating the output with the batch means and independent replications techniques, it was apparent that the latter's confidence intervals were too small to obtain good coverage of the mean value. This was due to the higher number of observations, which caused the confidence interval's width to become too tight around the biased mean.

Batch means provided better coverage of both the grand mean of the data and the analytic mean of 12, but these rates dropped as the transient cutoff value increased. This indicated that the system was still operating within some portion of the transient, with an initial high bias caused by the original filling of the queue.

Employing the Kalman filter resulted in a number of interesting findings. The requirement for equally spaced data, for the particular adaptation technique used to generate the adaptive Kalman filter, led to the mean being biased low. Elimination of the average bias produced 95% confidence interval coverage rates, of the analytic mean,

which were superior to both batch means and independent replications.

The greatest benefit, however, comes in the identification of the systems's true underlying variance. Because of the nature of the Kalman filter, the variance of the forecasts is a constant value once steady state has been attained. Using this value, along with the Kalman gain and state transition matrices, explained in Chapter II, the system's variance can be computed. This is accomplished with equations from the update cycle of the usual form or the innovations form of the Kalman filter, as described in Chapter II (18:176).

Another aspect of the Kalman filter which needs to be addressed deals with its primary purpose. It provides an excellent "step ahead" forecast for the next observation's characteristics, and this capability is currently available in software packages such as FORECAST MASTER.

Areas for Future Development

Several areas have come to light durirg the course of this thesis which would be interesting to follow-on research. They deal primarily with future research possibilities for the Kalman filter, and mainframe adaptation for the Output Analysis Program.

Harrison and Stevens (10:341-362) used Bayesian estimation to compute the probability that each new

observation is either from steady state, an outlier, a step change, or a slope change. The process effectively used an array of Kalman filters to compute these probabilities, assuming the system variance was known in advance. Extension of this theory, known as multiple model adaptive estimation (17:129,131), however, utilizing a bank of adaptive Kalman filters, could be directly applied to identifying the end of the simulation transient period. Once the system has reached steady state, to within a specified probability, the covariance kernel can be computed and estimation of the mean can be accomplished.

The requirement for equally spaced data caused many bias problems in the interpolated data. Three approaches to attempt removing this bias are presented. First, further research could address the impact of using the original data values directly from SLAM II, and what impact, if any, this has on the estimate of the system's mean.

The second technique is an extension of the concept of "moving average," presented in Chapter II (27:293). The problem with the current interpolation technique is that the interpolated points sample more often across the lower time-in-system, higher time-between-observation data points, and each is considered a valid data point. The impact of these points could be reduced by simply averaging the first five actual data values on each side of the interpolated value.

This parallels the moving average technique and should result in a much more accurate representation of the data.

The third approach deals with the exponential representation of the transition matrix, $F_t = e^{A\delta t}$ (23:68). Given this relationship, identification of the underlying matrix A would be represented as:

$$A\delta t = \ln (F) \tag{48}$$

which represents an infinite Taylor's series.

Once the A matrix is determined it will allow for non-fixed sample rates of the data, equating directly to the δt values between the actual observations. Individual F matrices, F_k, could then be generated for each of the observations using the equation:

$$F_k = e^{(A\delta t_k)} \tag{49}$$

Another area which came to light was to use of the Kalman predicted values, along with the original data, to form a tighter confidence interval on the estimator of the mean. In this way, the Kalman filter predictions would be treated as a second, and independent, representation of the system's steady state mean value.

Limitations were encountered in the upper bound of 2000 observations for FORECAST MASTER. Development of a program which incorporates adaptive filtering techniques for larger

files would be beneficial in determining filter values across larger data sets.

Along the same lines, the Output Analysis Program has certain file size limits, as it was designed for a microcomputer. The program could be converted to Pascal and placed on a mainframe computer, to alleviate these limitations.

Appendix A: SLAM II and FORTRAN Code for M/M/1 Queue

SLAM II Code:

```
GEN,CAPT PORTER,MM1QUEUE,1/2/89,5,,,,,,,72;
LIMITS,2,3,500;

NETWORK

        CREATE,EXPON(6,1),0,1,1000;

        QUEUE(1);
             ACT(1)/1,EXPON(4,2);

        EVENT,1;

        ENDNETWORK;
```

FORTRAN CODE:

```
      PROGRAM MAIN
      DIMENSION NSET(10000)
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)

      COMMON QSET(10000)
      EQUIVALENCE(NSET(1),QSET(1))
      NNSET=10000
      NCRDR=5
      NPRNT=6
      NTAPE=7

      OPEN(10,FILE='MONE.OUT',STATUS='OLD')
      CALL SLAM
      STOP
      END
C
      SUBROUTINE EVENT(I)
      COMMON/SCOM1/ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)


C     COMPUTE TIME BETWEEN ARRIVALS AND SEND TO FILE 10

      GOTO (1)I
1     XX(3)=TNOW-ATRIB(1)
      XX(2)=TNOW-XX(1)
      WRITE(10,10) XX(3),XX(2)
```

```
10    FORMAT(2F12.7)

      XX(1)=TNOW

      RETURN
      END
```

Appendix B: Output Analysis Program User's Guide

OUTPUT ANALYSIS PROGRAM

USER'S MANUAL

Developed by

Captain Charles H. Porter

OUTPUT ANALYSIS PROGRAM

Introduction:

The Output Analysis Program was developed to evaluate the mean and variance for simulation outputs using the techniques of independent replications and batch means. The program was developed to operate on sequential output files of uniform length, with one entry per line. Data must be in an ASCII format. As a note, computation of 95% confidence intervals uses a t-value rounded off to the nearest 1/10th.

There are four main topic areas within the program: the Data Menu, the Graph, Batch Means, and Independent Replications. This manual will address each individually, along with operating instructions.

Output of any desired screen may be obtained by pressing <SHIFT> Prt Scr, while a working printer is on line.

Program Operation:

The Output Analysis Program is menu driven. Selection of options on menus requires only the number of the selection to be pressed, that is with no carriage return. When the user is queried for file names and cutoff values a carriage return is required to end the entry. File names should be kept short, but have a maximum limit of 16 characters.

Initialization:

Data for the Output Analysis Program must be on the same computer disk drive, and in the same computer directory as the executing file, or have its appropriate drive\directory identified. For instance, if the program is on drive A: and the data is in the directory DATA, on drive C:, in a file named OUTPUT.DAT, then when queried for the file name the user must respond C:\DATA\OUTPUT.DAT. If the data is in the same directory as the program it is necessary only to type the filename and extension (OUTPUT.DAT). To accomplish this either the executable file OUTPUT.EXE can be copied onto the computer hard disk (if available) along with the data to be used, or OUTPUT.EXE can be copied onto a floppy diskette along with the data. As a note the .EXE file requires less than 60K of disk space.

To initialize the Output Analysis Program, place the program diskette into a 5 1/4 inch floppy disk drive, log into the appropriate drive and either copy the file OUTPUT.EXE to the hard disk or to a floppy containing the data file. Now, log into the appropriate drive and type "GRAPHICS." This will enable the graphic displays within the program to be printed. Then type "OUTPUT" . The program's Data Menu will now appear on the screen. Pressing the <shift> key and the Print Screen key together will now pass whatever is displayed on the screen to the printer.

Data Menu:

The Data Menu provides the user with four options.

1) BUILD NEW DATA DIRECTORY
2) RECALL EXISTING DIRECTORY
3) LIST EXISTING DIRECTORIES

0) EXIT

1) BUILD NEW DATA DIRECTORY

This option allows the user to construct a new data file from a simulation output file. Again, the data must be in sequential files of equal length. The user is queried for the name of the new directory to build, the number of successive outputs files to be read in, the length of these output files, and the name of the simulation output file. The values are then read from the specified simulation file into a file "FILENAME.dct". See below for example.

EXAMPLE 1. (Computer Queries are in **BOLD**)

SIM.OUT (Consists of two simulations outputs of five values each, obtained from a program such as SLAM2)

1
2
3
4
5
2
3
4
4
6

NAME OF NEW DIRECTORY: EXAMPLE

NO. OF FILES IN DIRECTORY: 2

NO. OF ENTRIES PER FILE: 5

NAME OF FILE: SIM.OUT

DIRECTORY EXAMPLE.DCT IS NOW BUILT

At this point the data has been loaded into a .DCT file and can be evaluated by the graph options listed in the next section.

If the user accidently input SUM.OUT for the name of the simulation output data file (rather than SIM.OUT), the program would return the appropriate error message that the file did not exist. It then provides the user the choice of reentering the file name or returning to the Data Menu.

2) RECALL EXISTING DIRECTORY

This option is used if the data file required was loaded previously, and the name is known. The user will be prompted for the directory name (NOTE: The .DCT extension is assumed and not entered). The program will then provide the user with the number of files and number of entries per file for the directory which was recalled. The user then has the option to choose one of the two graph options described in the GRAPH section below, or to return to the Data Menu and specify a different file for use.

If a file is requested which does not exist then the program will return an error message indicating the file could not be found. The user is then given the choice of trying to load a file again or returning to the Data Menu.

3) LIST EXISTING DIRECTORIES:

Like the Recall option, this option allows the user to retrieve data directories previously created. When selected, however, it provides a list of all .DCT files currently on disk. The user is then prompted to either select one of these files for use or return to the main Data Menu. If the user wants to load one of the listed files the recall routine is automatically called and the commands are the same.

0) EXIT:

This selection terminates the Output Analysis Program and returns the user to the DOS shell. All steady state files (discussed in the Graph Section) and .DCT files generated during the session are automatically saved.

GRAPH:

The Graph portion of the program incorporates a number of data techniques and operations. When the Data portion of the program terminates, either when building a new directory or recalling an old one, the user will have the following two options.

1) IDENTIFY TRANSIENT, MULTIPLE FILES

2) IDENTIFY TRANSIENT, SINGLE FILE

The selection criteria are self explanatory. If the user is evaluating multiple runs, planning to use the Independent Replications option, then option 1 is selected. If, on the other hand, he/she is working with one long run, for Batch Means, then option 2 is selected.

1) IDENTIFY TRANSIENT, MULTIPLE FILES:

In option 1, the multiple runs are averaged across the individual observation numbers. For instance, the first value of file one is averaged with the first value of file 2, and file 3, etc. These average values are then displayed on the computers screen, along with a scale. The user must then interpret, from the display, the point at which the transient period ends. The user is allowed to scroll through all of the input data by typing a "1" to back-up one screen, and a "2" to go forward one. When the user has determined the extent of the transient he/she types the number 3 and is queried to enter the appropriate transient "cutoff" value, and answer yes or no (Y/N) to having a copy of the current graph screen printed..

2) IDENTIFY TRANSIENT, SINGLE FILE:

In option 2, the data is "smoothed" by running a moving average across the data. Reference Peter D. Welch's work in the Computer Perfomance Modelling Handbook (27:293). This is accomplished by averaging values within a "window" whose

width is specified by the user at the prompt "**SIZE OF MOVAVG WINDOW ?**", and plotting this average as the value at the center of the window. This window moves to the right, one observation at a time, until there isn't enough data to fill the window. This smoothed data is then graphed on the screen, and the user has the same options as noted above to scroll through the data. After the "Cutoff" value is input, and the user has answered whether he/she would like a print out, the cutoff number is used to develop a file of steady state values.

STEADY STATE FILE GENERATION:

Once the transient "cutoff" value has be attained, the program edits the current .DCT file and truncates the first "cutoff" number of entries from each of the files. The user is asked for the name of the new Steady State File, to which the program adds a .SST extension. This file will be the one used to compute the mean and variance for the system while in steady state.

When the .SST file has been created, the program will ask the user to select either the Batch Means or Independent Replications techniques. Selection is based on the number of files within the .SST file.

BATCH MEANS:

The Batch Means portion of the program evaluates one long run of the simulation. The data within the .SST file is partitioned into larger and larger batches until independence, between the batches, is obtained. The algorithm used was developed by Fishman (8:510-511). When the test for independence passes, the number of batches required for independence, the estimated mean and variance, and the 95% confidence interval (assuming a t-value of 1.96) are displayed on the screen.

If independence is not obtained before the number of batches decreases to less than eight (8), then the message "TEST FOR INDEPENDENCE FAILED, NEED MORE DATA", will appear on the screen. Typing a carriage return will now return the user to the initial Data Menu, from which he/she can accomplish further runs or exit.

Independent Replications:

The Independent Replications procedure also uses the truncated .SST data file. The difference is, of course, that there are multiple simulation outputs in the .SST file used by the Independent Replications procedure.

The procedure automatically calculates the 95% confidence interval for the data, using the algorithms noted by Welch in the Computer Performance Modelling Handbook (27:280). The mean, variance, confidence interval, and number of additional runs required to be within +/- 5% of the computed mean are displayed on the screen for the user's information. The additional runs required are computed with the formula identified by Welch, for Pilot Experimentation (27:320).
After the data are displayed, typing carriage return will return the user to the initial data menu, to accomplish further runs or to exit.

Appendix C: Output Analysis Program

```
{$M 65520,0,65519}    {Sets Heap Size to Maximum Allowable}
PROGRAM SIMALYSIS;
{This program contains the graphic lead in slide for the Output
 Analysis Program and a call to the unit PROC, which contains the
 procedures necessary to do the calculations.  No actual computations
 or data activities take place within the main program}

USES CRT,DOS,PROC,GRAPH;    {SPECIFY THE UNITS THE PROGRAM WILL USE}

PROCEDURE LEADIN;
{This procedure simply produces the introductory screen for the
 Output Analysis Program}

VAR
   GD,GM,ERRCODE    :INTEGER;

BEGIN
   READLN;                      {GRAPHICS LEAD IN SLIDE GENERATION}
   GD:=DETECT;                  {FROM EXAMPLE IN TURBO PASCAL REF. GUIDE}
   INITGRAPH(GD,GM,'');
   ERRCODE:=GRAPHRESULT;
   IF ERRCODE = GROK THEN
   BEGIN
      SETTEXTSTYLE(TRIPLEXFONT,HORIZDIR,5);
      SETBKCOLOR(BLUE);
      TEXTCOLOR(YELLOW);
      OUTTEXTXY(35,100,'OUTPUT ANALYSIS PROGRAM');
      SETTEXTSTYLE(DEFAULTFONT,HORIZDIR,1);
      OUTTEXTXY(400,240,'Developed by');
      OUTTEXTXY(30,310,'Press <RETURN> to begin');
      SETTEXTSTYLE(TRIPLEXFONT,HORIZDIR,1);
      OUTTEXTXY(400,260,'Capt Charles H. Porter');
      READLN;
   END

   ELSE                      {IN CASE OF GRAPHICS ERROR}
      WRITELN('GRAPHICS ERROR ;',GRAPHERRORMSG(ERRCODE));

   CLOSEGRAPH;
   TEXTMODE(C80);
END;
(* MAIN MENU *)
BEGIN
   CLRSCR;
   LEADIN;                  {DISPLAYS LEAD IN GRAPHICS FROM ABOVE}
   DATA(TEMP,NUMFILE,NUMENT); {INITIAL CALL TO UNIT PROC}
   CLRSCR;
END.
```

```
UNIT PROC;

{THIS UNIT IS COMPRISED OF ALL THE PROCEDURES NECESSARY TO PERFORM THE
OUTPUT ANALYSIS TECHNIQUES OF BATCH MEANS AND INDEPENDENT REPLICATIONS.
IT IS ACCESSED BY THE PROGRAM SIMALYSIS TO PERFORM THESE FUNCTIONS}


INTERFACE
USES CRT,DOS,GRAPH,GRAPH3;

TYPE ARA = ARRAY [0..2000] OF REAL;
     WRI = STRING[20];
     FYLE= TEXT;

VAR TEMP,TEMPP,MAVG        :ARA;        {GLOBAL VARIABLE DECLARATION}
    NUMENT,NUMFILE         :INTEGER;
    PAUSE                  :CHAR;
    FILNAME,DCTNAME        :WRI;
    FIL                    :FYLE;
    ORDER,CUTOFF,NUM       :INTEGER;


PROCEDURE PRTSC;          {PROCEDURE DECLARATION IN THE INTERFACE SECTION}
PROCEDURE BOXES;
PROCEDURE STSTATE;
PROCEDURE BATCH;
PROCEDURE SST;
PROCEDURE GRAPHIT;
PROCEDURE AVERAGE;
PROCEDURE MOVAVG;
PROCEDURE NEWDATA;
PROCEDURE VIEW;
PROCEDURE DATA(TEMP:ARA;VAR NUMFILE,NUMENT:INTEGER);


IMPLEMENTATION
{THIS PORTION OF THE UNIT DOES THE ACTUAL OPERATIONS ON THE
PROCEDURES/FUNCTIONS IDENTIFIED IN THE INTERFACE SECTION}
```

```
FUNCTION EXIST(FILENAME:STRING):BOOLEAN;
{This function identifies whether a file actually exists.
 It is used called just before a file is to be accessed.
 If the file does not exist the user is given the opportunity
 to try again, or back up.  If this function were not included
 then the program would abort whenever an action was attempted
 on a non-existent file}

VAR
   FIL   :FILE;
BEGIN
   ASSIGN(FIL,FILENAME);
   {$I-}
   RESET (FIL);
   {$I+}
   EXIST := (IORESULT = 0);
END;

PROCEDURE PRTSC;
{THIS PROCEDURE PRINTS DATA CURRENTLY DISPLAYED ON SCREEN.  IT IS
ACCESSED BY THE GRAPHIT PROCEDURE TO GET A "HARD COPY" OF THE
ACTUAL DATA PLOT}

VAR REG:REGISTERS;
BEGIN
INTR($5,Dos.Registers(REG))
END;

PROCEDURE BOXES;
{This procedure draws boxes on the screen, within which
 the selection rules are placed, in menu format.}

TYPE
   bar = string[79];

var
  lyne : bar;
  UL,vl,ur,ll,lr,li,ri,mi:char;
  X:INTEGER;
begin
TEXTCOLOR(WHITE);
UR:=CHAR(187);
LR:=CHAR(188);
LL:=CHAR(200);
LI:=CHAR(185);
RI:=CHAR(204);
VL:=CHAR(186);
UL:=CHAR(201);
LYNE:='═══════════════════════════════════════════════════════════════';
CLRSCR;
```

```
GOTOXY(12,5);   WRITE(UL);
GOTOXY(62,5);   WRITE(UR);
GOTOXY(13,5); WRITE(LYNE);
GOTOXY(13,7); WRITE(LYNE);
GOTOXY(13,18);WRITE(LYNE);
GOTOXY(13,20);WRITE(LYNE);
FOR X:= 6 TO 19 DO
BEGIN
  GOTOXY(12,X); WRITE(VL);
  GOTOXY(62,X); WRITE(VL);
END;

GOTOXY(12,20);  WRITE(LL);
GOTOXY(62,20);  WRITE(LR);
GOTOXY(12,18);  write(ri);
GOTOXY(62,18);  write(li);
GOTOXY(12,7);   write(ri);
GOTOXY(62,7);   write(li);
TEXTCOLOR(YELLOW);
END;

PROCEDURE STSTATE;

{This procedure evaluates the data in a steady state directory
 which was developed in the SST procedure.  It uses the technique of
 independent replications and calculates the confidence interval for the
 data provided.  A T-value of 1.96, corresponding to a 95% confidence
 interval for large amounts of data, will be used.}

VAR
I,J,K,M                 :INTEGER;
MU,TOTAL,SQDIF          :ARA;
MUHAT,LOW,HIGH,T        :REAL;
SSQ,S,MUTOT,SUM,DIF     :REAL;
PAUSE                   :CHAR;
TVAL                    :STRING[4];
NUMRUN,ADDRUN           :INTEGER;

BEGIN
   CLRSCR;
   BOXES;
   TEXTCOLOR(YELLOW);
   GOTOXY(25,6); WRITE('STEADY STATE ANALYSIS');

   M:=1;                                        {INITIALIZES COUNTER FOR
DATA RETRIEVAL}

   MUTOT:=0;                                    {INITIALIZE VARIABLES TO
ZERO}
   MUHAT:=0;
```

```
   FOR I:=1 TO NUMFILE DO
      TOTAL[I]:=0;


   FOR I:=1 TO NUMFILE DO                    {COMPUTE MEANS OF INDIVIDUAL
FILES}
   BEGIN
      FOR J:=1 TO NUM DO
      BEGIN
         TOTAL[I]:=TOTAL[I]+TEMPP[M];
         M:=M+1;
      END;
      MU[I]:=TOTAL[I]/NUM;

      MUTOT:=MUTOT+MU[I];                    {SUM ALL OF THE INDIVIDUAL
MEANS}

   END;

   MUHAT:=MUTOT/NUMFILE;                     {COMPUTE THE GRAND MEAN}

   SUM:=0;

   FOR K:=1 TO NUMFILE DO                 {CALCULATE THE VARIANCE OF THE
DATA}
   BEGIN
      SQDIF[K]:=SQR(MU[K]-MUHAT);
      SUM:=SUM+SQDIF[K];
   END;
   SSQ:=SUM/(NUMFILE-1);

   S:=SQRT(SSQ);                             {OBTAIN THE STANDARD
DEVIATION}

   IF NUMFILE=1 THEN                         {SET THE APPROPRIATE
T-VALUE}
      T:=12.71;
   IF NUMFILE=2 THEN
      T:=4.303;
   IF NUMFILE=3 THEN
      T:=3.182;
   IF NUMFILE=4 THEN
      T:=2.776;
   IF NUMFILE=5 THEN
      T:=2.571;
   IF (NUMFILE=6) OR (NUMFILE=7) THEN
      T:=2.4;
   IF (NUMFILE=8) OR (NUMFILE=9) THEN
      T:=2.3;
   IF (NUMFILE>=10) AND (NUMFILE<=13) THEN
```

```
      T:=2.2;
   IF (NUMFILE>=14) AND (NUMFILE<=27) THEN
      T:=2.1;
   IF (NUMFILE>=28) AND (NUMFILE<=120) THEN
      T:=2.0;
   IF NUMFILE>120 THEN
      T:=1.96;


   STR(T,TVAL);
   GOTOXY(17,10); WRITE('95% CONFIDENCE INT COMPUTED WITH T=',TVAL);

   LOW:=(MUHAT-((T*S)/SQRT(NUMFILE)));          {COMPUTE THE 95% CONF
INT}
   HIGH:=(MUHAT+((T*S)/SQRT(NUMFILE)));

   GOTOXY(22,12); WRITE('EST. MEAN : ',MUHAT:6:2);       {DISPLAY
SOLUTIONS}
   GOTOXY(22,13); WRITE('VARIANCE : ',SSQ:6:2);

   GOTOXY(21,15); WRITE(LOW:9:2,' <= MEAN <=  ',HIGH:6:2);

   {The variable ADDRUN represents the number of additional runs of
    the simulation required to obtain a confidence interval which
    is within +/- 5% of the mean.  Reference Welch, in Chapter 6
    of the Computer Performance Modeling Handbook, pages 321-322}


   DIF:=(HIGH-LOW);
   NUMRUN:=TRUNC((DIF/(MUHAT/10))*NUMENT);
   ADDRUN:=NUMRUN-NUMENT;
   IF ADDRUN<0 THEN
      ADDRUN:=0;

   GOTOXY(14,17);WRITE('ADDITIONAL RUNS REQ FOR +/- 5% INTERVAL =
',ADDRUN);

   PAUSE:=READKEY;
END;


PROCEDURE BATCH;
{This procedure utilizes Fishman's algorithm for computing independence
between the batches.  It starts with a batch size of one, and then
doubles it until either independence is obtained or the number of
batches
decreases to less than 8.  Specifics on the algorithm can be found in
Fishman's article, "Grouping Observations in Digital Simulation",
Management Science,24,5,1978, pp 510-521.  The intent of the algorithm
is to compute system mean and variance from one long run.}
```

```
VAR
   BREAK                    :LONGINT;    (NUMBER OF BATCHES)
   D,I,J,M,N                :INTEGER;
   A,B,VARC,MTOT,K,Z        :REAL;
   TTOP,TBOTT,MEAN,T        :REAL;
   TOT,YAVG,CM,LOW,HIGH     :REAL;
   MU                       :ARA;
   SUMSQR,MEANSQR,SKSQ      :REAL;
   SK,HW                    :REAL;


BEGIN

IF NUMFILE=1 THEN                    (IF THE FILE IS APPROPRIATE PROCEED)
  BEGIN
     TOT:=0;                                    (INITIALIZE VARIABLES)
     Z:=0;
     CLRSCR;
     BOXES;
     TEXTCOLOR(YELLOW);
     GOTOXY(28,6);WRITE('BATCH MEANS');
     FOR I:=1 TO NUM DO                         (SUM THE STEADY STATE DATA)
        TOT:=TOT+TEMPP[I];

      YAVG:=TOT/NUM;                            (COMPUTE THE AVERAGE OBS
VALUE)
         MTOT:=0;
         D:=0;
         K:=NUM;
         N:=1;



         WHILE K >= 8 DO                        (WHILE NO. OF BATCHES
REMAINS >8)
         BEGIN                                  (PERFORM THE BATCH MEANS
TEST)
            TTOP:=0;
            TBOTT:=0;
            J:=1;
            Z:=1.96;
            BREAK:=TRUNC(NUM/N);                (COMPUTE NUMBER OF BATCHES)

            FOR I:=1 TO BREAK DO                (FOR EACH BATCH)
            BEGIN
               TOT:=0;
               FOR M:=1 TO N DO                 (SUM THE NUMBER OF
ENTITIES/BATCH)
```

```
            BEGIN
               TOT:=TOT+TEMPP[J];
               J:=J+1;
            END;
            MU[I]:=TOT/N;                    {COMPUTE EACH BATCH'S MEAN}
         END;

         FOR I:=1 TO (BREAK-1) DO            {COMPUTE THE TEST
STATISTICS}
         BEGIN
            A:=((MU[I]-MU[I+1])*(MU[I]-MU[I+1]));
            TTOP:=TTOP+A;
            B:=(MU[I]-YAVG)*(MU[I]-YAVG);
            TBOTT:=TBOTT+B;
         END;
         TBOTT:=TBOTT+((MU[BREAK]-YAVG)*(MU[BREAK]-YAVG));
         CM:= 1-(TTOP/(2*TBOTT));

         VARC:=ABS((BREAK-2)/((BREAK*BREAK)-1));   {COMPUTE VARIANCE}

         IF CM > (Z*SQRT(VARC)) THEN    {SEE IF STATISTIC CM PASSES
TEST}
         BEGIN
            N:=2*N;                    {IF NOT, INCREMENT N AND K AND
RETRY}
            K:=K/2;
         END;

         SUMSQR:=0;
         MTOT:=0;


         IF CM < (Z*SQRT(VARC)) THEN    {IF CM DOES PASS THE TEST}
         BEGIN
            D:=N;                          {INDEPENDENCE HAS BEEN
OBTAINED}
            K:=0;
            FOR I:=1 TO BREAK DO
            BEGIN
               MTOT:=MTOT+MU[I];           {CALCULATE THE GRAND MEAN}
               SUMSQR:=SUMSQR+SQR(MU[I]);
            END;
                                           {FIND APPROPRIATE T-VALUE}
            IF NUMFILE=1 THEN
               T:=12.71;
            IF NUMFILE=2 THEN
               T:=4.303;
            IF NUMFILE=3 THEN
               T:=3.182;
            IF NUMFILE=4 THEN
```

```
                T:=2.776;
             IF NUMFILE=5 THEN
                T:=2.571;
             IF (NUMFILE=6) OR (NUMFILE=7) THEN
                T:=2.4;
             IF (NUMFILE=8) OR (NUMFILE=9) THEN
                T:=2.3;
             IF (NUMFILE>=10) OR (NUMFILE<=13) THEN
                T:=2.2;
             IF (NUMFILE>=14) OR (NUMFILE<=27) THEN
                T:=2.1;
             IF (NUMFILE>=28) OR (NUMFILE<=120) THEN
                T:=2.0;
             IF NUMFILE>120 THEN
                T:=1.96;


             MEAN:=MTOT/BREAK;              {COMPUTE MEAN AND VARIANCE}
             MEANSQR:=SQR(MEAN);
             SKSQ:=((SUMSQR-(BREAK*MEANSQR))/(BREAK-1));
             SK:=SQRT(SKSQ);
             HW:=T*(SK/(SQRT(BREAK)));     {COMPUTE CONF INT HALFWIDTH}

             GOTOXY(23,9);WRITE('INDEP WITH ',BREAK,' BATCHES');
             GOTOXY(23,11);WRITE('EST. MEAN: ',MEAN:6:2);
             GOTOXY(23,12);WRITE('VARIANCE: ',SKSQ:6:2);

             LOW:=MEAN-HW;     {COMPUTE CONFIDENCE INT}
             HIGH:=MEAN+HW;
             GOTOXY(21,15);WRITE(LOW:9:2,' <= MEAN <= ',HIGH:6:2);

          END;
       END;

    {GIVEN THAT INDEPENDENCE IS NOT OBTAINED BEFORE THE NUMBER OF
ENTITIES
     PER BATCH FALLS BELOW 8, THEN THE FOLLOWING MESSAGE IS DISPLAYED}

    IF D=0 THEN
    BEGIN
       GOTOXY(17,12);WRITELN('TEST FOR INDEPENDENCE FAILED, NEED MORE
DATA');
    END;
    READLN;
  END;
END;

PROCEDURE SST;

{This procedure is called after the user has seen a graphical
```

```
 representation of the data, with confidence intervals, and has
 an idea where the transient period terminates.  The program
 queries the user for this point, and then modifies the current
 .dct directory to build a .sst directory of steady state data.
 After the directory is built the STSTATE and Batch procedures
 are called to obtain the data's confidence interval.}

VAR
  SSTNAME         :WRI;
  SST             :FYLE;
  I,J,M,N,GO         :INTEGER;
  PAUSE        :CHAR;

BEGIN
   CLRSCR;
   TEXTCOLOR(WHITE);
   BOXES;
   TEXTCOLOR(YELLOW);
   GOTOXY(21,6);WRITE('CREATE STEADY STATE DIRECTORY');

   IF TEMP[1]=0 THEN              {CHECK TO MAKE SURE A DIRECTORY IS
LOADED}
   BEGIN
      GOTOXY(17,10);WRITE('NO DIRECTORY IN USE');
      GOTOXY(17,12);WRITE('PLEASE PRESS <RETURN> FOR MAIN MENU');
      PAUSE:=READKEY;
   END;

   IF TEMP[1]<>0 THEN             {GIVEN THAT A DATA FILE IS LOADED}
   BEGIN

   {At this point the program will echo the cutoff value loaded in the
   GRAPHIT procedure.  The user will then be queried for the name of
   the Steady State file which will be generated by truncating the first
   "cutoff" number of values from each sequential file.  The program
will
   take the user's input name and add the extension ".SST" }

      GOTOXY(17,9);WRITE('CUTOFF VALUE IS ',CUTOFF);
      GOTOXY(17,10);WRITE('NAME OF STEADY STATE DIRECTORY: ');
      GOTOXY(18,11);READLN(SSTNAME);
      SSTNAME:=SSTNAME+'.SST';          {ADD THE .SST EXTENSION TO
FILENAME}
      ASSIGN(SST,SSTNAME);
      REWRITE(SST);                        {OPEN THE FILE FOR DATA}
      NUM:=NUMENT-CUTOFF;               {DEFINE THE NUMBER OF VALUES PER
FILE}

      WRITELN(SST,NUMFILE);         {MAKE THE NUMBER OF FILES THE FIRST
ENTRY}
```

```
      WRITELN(SST,NUM);          {MAKE THE NUMBER OF VALUES PER FILE,
SECOND}
      M:=1; N:=1;
      FOR I:=1 TO NUMFILE DO      {READ IN THE VALUES FROM THE ORIGINAL
DATA}
      BEGIN                       {SET, TRUNCATING THE FIRST "CUTOFF"
VALUES}
         FOR J:=1 TO NUMENT DO
         BEGIN
            IF J<CUTOFF THEN
               M:=M+1;


            IF J>=CUTOFF THEN
            BEGIN
               WRITELN(SST,TEMP[M]);
               TEMPP[N]:=TEMP[M];
               N:=N+1;
               M:=M+1;
            END;
         END;
      END;
   CLOSE(SST);                    {CLOSE THE NEW .SST FILE}
   CLRSCR;
   BOXES;

   {Now that the steady state data file is built, determine if the
    user wants to evaluate the new file via Batch Means (one long run)
    or Independent Replications (Multiple runs)}

   GOTOXY(21,6);WRITE('CREATE STEADY STATE DIRECTORY');
   GOTOXY(17,10);WRITE('DIRECTORY ',SSTNAME,' HAS BEEN CREATED');
   GOTOXY(17,12);WRITE(' 1)  INDEPENDENT REPLICATIONS');
   GOTOXY(17,13);WRITE(' 2)  BATCH MEANS EVALUATION');
   GOTOXY(17,14);WRITE(' 3)  DATA MENU');
   GOTOXY(17,19);WRITE('PLEASE ENTER NUMBER OF CHOICE : ');
   READLN(GO);

   IF (GO=1) AND (NUMFILE=1) THEN
   BEGIN
      TEXTCOLOR(GREEN);
      GOTOXY(17,16);WRITE('MUST USE BATCH MEANS FOR SINGLE DATA FILE');
      GOTOXY(17,17);WRITE('PRESS <RETURN> FOR BATCH MEANS PROCESS.');
      TEXTCOLOR(YELLOW);
      READLN;
      GO:=2;
   END;

   IF (GO=2) AND (NUMFILE>1) THEN
   BEGIN
```

```
      TEXTCOLOR(GREEN);
      GOTOXY(17,16);WRITE('MUST USE IND REPLICATIONS FOR MULTIPLE
FILES');
      GOTOXY(17,17);WRITE('PRESS <RETURN> FOR IND REP PROCESS.');
      TEXTCOLOR(YELLOW);
      READLN;
      GO:=1;
   END;

   CASE GO OF
       1 : STSTATE;
       2 : BATCH;
      END;
   END;
END;



PROCEDURE GRAPHIT;

{This procedure graphs the data in the current .dct directory for the
 user to review.  It then calls to the SST procedure to build a
directory
 for the steady state data.}

type sett = array [1..101,1..2] of real;
var
    PRYNT,pause :CHAR;
    GraphDriver : integer;  { The Graphics device driver }
    GraphMode   : integer;  { The Graphics mode value }
    ErrorCode   : integer;  { Reports any graphics errors }
    MaxColor    : word;     { The maximum color value available }
    OldExitProc : Pointer;  { Saves exit procedure address }
    step        : INTEGER;
    values      : sett;
    x,Y,MAXADD,VAL,YY    : integer;
    max,scale            : real;
    WRITING              : WRI;
    YAXIS,I,START,STOP   : INTEGER;
    VALUE,YYY            : STRING[5];
    CHOICE,SER,NUMGRPH   : INTEGER;

BEGIN

STEP:=1;
IF NUMENT<600 THEN
   STEP:=ROUND(600/NUMENT);
 MAX :=0;
 FOR X:= 1 TO NUMENT DO
   IF (MAVG[X]>MAX) OR (MAVG[X]*-1>MAX) THEN BEGIN
```

```
    MAX:= MAVG[X];
    MAXADD:=X;
END;
IF MAX<0 THEN MAX:=MAX*-1;
IF MAX=0 THEN EXIT;
graphdriver:=2;
graphmode:=4;
InitGraph(graphdriver,graphmode,'');  { activate graphics }
ErrorCode := GraphResult;               { error? }
if ErrorCode <> grOk then
begin
  Writeln('Graphics error: ', GraphErrorMsg(ErrorCode));
  Halt(1);
end;

CHOICE:=1;
SER:=1;
NUMGRPH:=(TRUNC(NUMENT/600)+1);
START:=1;



WHILE CHOICE <> 3 DO
BEGIN
   CLEARDEVICE;
   IF SER=0 THEN          {ERROR MESSAGE IF USER TRIES TO BACK UP}
   BEGIN                     {WHEN STARTING PT IS ON SCREEN}
      SER:=1;
      START:=1;
      OUTTEXTXY(17,165,'STARTING POINT IS ALREADY ON SCREEN');
   END;
   IF SER>NUMGRPH THEN   {ERROR MESSAGE IF USER TRIES TO GO FORWARD}
   BEGIN                        {WHEN ENDING PT IS ON SCREEN}
      SER:=NUMGRPH;
      START:=START-600;
      OUTTEXTXY(17,165,'END OF OUTPUT DATA IS ON SCREEN');
   END;

   LINE(25,140,625,140);      {DRAW AXES AND TICK MARKS}
   LINE(25,40,25,140);

   LINE(85,140,85,143);
   LINE(145,140,145,143);
   LINE(205,140,205,143);
   LINE(265,140,265,143);
   LINE(325,140,325,143);
   LINE(385,140,385,143);
   LINE(445,140,445,143);
   LINE(505,140,505,143);
```

```
    LINE(565,140,565,143);
    LINE(625,140,625,143);

    FOR I:=1 TO 10 DO          {NUMBER THE Y-AXIS}
    BEGIN
       YY:=TRUNC((100-(10*(I-1)))*(MAX/100));
       STR(YY,YYY);
       OUTTEXTXY(5,(40+((I-1)*10)),YYY);
    END;



    FOR I:=1 TO 10 DO          {NUMBER THE X-AXIS}
    BEGIN
       VAL:=((START-1)+(TRUNC((I*60)/STEP)));
       STR(VAL,VALUE);
       OUTTEXTXY((23+(I*60)),147,VALUE);
    END;


    MOVETO(25,140);          {DRAW THE 600 DATA PTS STARTING WITH
(START-1)}
    STOP:=(600+(START 1));
    IF STOP>NUMENT THEN
       STOP:=NUMENT;
    FOR Y:= START TO STOP DO
       LINETO(25+((Y-(START-1))*STEP),130-round((MAVG[Y]/max)*100));
    OUTTEXTXY(17,155,'1)LAST SCREEN  2)NEXT SCREEN  3)ID TRANSIENT');

    GOTOXY(50,21);READLN(CHOICE);   {ALLOWS USER TO MOVE THROUGH THE
DATA}

    IF CHOICE=1 THEN                {RESET STARTING PT ACCORDINGLY}
    BEGIN
       SER:=SER-1;
       START:=START-600;
    END;
    IF CHOICE=2 THEN
    BEGIN
       SER:=SER+1;
       START:=START+600;
    END;
 END;

 {When the user determines a "cutoff" value for the transient, based on
  the graph he/she types "3" and is queried for the cutoff value.}

 OUTTEXTXY(17,175,'WHERE DO YOU THINK TRANSIENT ENDS? :');
 GOTOXY(50,23);
 READLN(CUTOFF);
```

```
 {If the user would like, the graph can be dumped to the printer}

 OUTTEXTXY(17,185,'WOULD YOU LIKE A PRINTOUT? (Y/N): ');PRYNT:=READKEY;

 IF UPCASE(PRYNT)='Y' THEN
   PRTSC;
 TEXTMODE(C80);
 SST;
END;

PROCEDURE AVERAGE;

{This procedure can be accessed from the MAIN MENU or from the
 RECALL procedure.  It averages the values across replications and
 then performs a Moving Average algorithm with the moving window
 size determined by the user.  This data is then forwarded to the
 GRAPHIT procedure to be displayed and evaluated.}

VAR
I,J,K              :INTEGER;
TOT                :ARA;

BEGIN
   FOR I:=1 TO NUMENT DO          {INITIALIZE ALL ARRAY VALUES TO ZERO}
   BEGIN
     TOT[I]:=0;
     MAVG[I]:=0;
   END;
   FOR I:=1 TO NUMENT DO          {FOR EACH ENTRY IN THE OUTPUT FILE}
   BEGIN
      K:=I;

      FOR J:=1 TO (NUMFILE) DO  {FOR EACH OUTPUT FILE}
      BEGIN
         TOT[I]:=TOT[I]+TEMP[K];      {COMPUTE A TOTAL VALUE FOR THE ITH
VALUE}
         K:=(J*NUMENT)+I;             {INCREMENT THE ORIGINAL FILE
COUNTER}
      END;
      MAVG[I]:=TOT[I]/NUMFILE;        {COMPUTE THE AVERAGE OF ALL ITH
VALUES}
   END;
   GRAPHIT;                           {GRAPH THE AVERAGE VALUES}
END;

PROCEDURE MOVAVG;
{This procedure parallels the AVERAGE procedure, but is used when there
is only
 one output file in the directory.  The procedure uses the Moving
```

```
Average
 algorithm noted by Welch in the "Computer Performance Modelling
Handbook",
 averaging across the specified "window" to smooth the data.  The data
 generated will be forwarded to the GRAPHIT procedure for display}


VAR
J,M,N              :INTEGER;
TOT                :ARA;
WINDOW             :INTEGER;          {HALF WIDTH OF MOVING WINDOW}
BEGIN
      BOXES;
      TEXTCOLOR(YELLOW);
      GOTOXY(30,6); WRITE('MOVING AVERAGE ');
      GOTOXY(17,10);WRITE(' NO. OF FILES IN DIRECTORY: ',NUMFILE);
      GOTOXY(17,12);WRITE(' NO. OF ENTRIES PER FILE: ',NUMENT);

{QUERY THE USER FOR THE MOVING AVERAGE WINDOW SIZE}
      GOTOXY(17,14);WRITE('HALFWIDTH OF MOVAVG WINDOW
?');READLN(WINDOW);

      FOR N:=1 TO NUMENT DO

      TOT[N]:=0;                  {INITIALIZE ALL TOTALS TO ZERO}

      FOR N:=1 TO NUMENT DO       {RUN THE WINDOW ACROSS ALL ENTITIES}
      BEGIN
         IF NK(WINDOW+1) THEN     {FOR THOSE BEFORE THE WINDOW HALF WIDTH}
         BEGIN
            FOR M:=1 TO ((2*N)-1) DO
            BEGIN
               TOT[N]:=TOT[N]+TEMP[M];
            END;
         MAVG[N]:=(1/((2*N)-1))*TOT[N];
         END;

         IF N>=(WINDOW+1) THEN
         BEGIN

         {After the entry is large enough to permit the full window
          to be used in the average the first algorithm below is used.
          This algorithm is used until the values do not allow for
          a full window size, due to the end of the file.  Then
          the second algorithm below is used}

            IF NK(NUMENT-WINDOW) THEN          {FIRST ALGORITHM}
            BEGIN
               FOR M:=(N-WINDOW) TO (N+WINDOW) DO
               TOT[N]:=TOT[N]+TEMP[M];
```

```
            MAVG[N]:=(1/((2*WINDOW)+1))*TOT[N];
          END;

          IF N>=(NUMENT-WINDOW) THEN          (SECOND ALGORITHM)
          BEGIN
            J:=(NUMENT-N);
            FOR M:=(N-J) TO NUMENT DO
              TOT[N]:=TOT[N]+TEMP[M];
            MAVG[N]:=TOT[N]/((2*J)+1);
          END;
      END;
   END;
      GRAPHIT;                          (GRAPH THE MOVING AVERAGES)

END;

PROCEDURE NEWDATA;

(This procedure is called from the DATA MENU to develop a
 new directory for use by the program.  It builds the directory
 based on user inputs for the number of files to be input (NUMFILE)
 and their common length of entries (NUMENT).  Inputs are prompted
 by the program.  Upon completion the user selects to evaluate the
 transient for either single or multiple output files, as appropriate.)

VAR I,J,M              :INTEGER;
    FILNAME,D          :WRI;
    FIL,DCT            :FYLE;
    GO                 :CHAR;
    TEST,TESST         :INTEGER;


    BEGIN
      BOXES;
      TEXTCOLOR(YELLOW);

      GOTOXY(30,6); WRITE('BUILD NEW DIRECTORY  ');
      GOTOXY(17,10);WRITE(' NAME OF NEW DIRECTORY:');READLN(DCTNAME);

      DCTNAME:=DCTNAME+'.DCT';        (TAKE THE USERS INPUT AND ADD .DCT)

      (At this point the user is queried for the number of simulation
       run outputs within the file and their length)

      GOTOXY(17,12);WRITE(' NO. OF FILES IN DIRECTORY:
');READLN(NUMFILE);
      GOTOXY(17,14);WRITE(' NO. OF ENTRIES PER FILE: ');READLN(NUMENT);


      ASSIGN(DCT,DCTNAME);       (ASSIGN A STRING DESIGNATOR TO THE
```

```
ARRAY)

      REWRITE(DCT);              (OPEN THE NEW DIRECTORY FOR USE)

      WRITELN(DCT,NUMFILE);      (FIRST ENTRY IS THE NUMBER OF OUTPUT
FILES)
      WRITELN(DCT,NUMENT);       (SECOND ENTRY IS THE NO. OF ENTITIES PER
RUN)

      M:=1;

      TEST:=1;
      TESST:=4;
      WHILE TEST=1 DO
      BEGIN
         CLRSCR;
         BOXES;
         TEST:=2;

         GOTOXY(30,6);WRITE('  FILE ENTRY');

         (Now the simulation output file to be read from is specified)

         GOTOXY(17,10);WRITE('NAME OF FILE : ');READLN(FILNAME);

         ASSIGN(FIL,FILNAME);  (ASSIGN THE STRING "FIL" TO THE ORIGINAL
FILE)

         IF (EXIST(FILNAME)=FALSE) THEN
         BEGIN
            GOTOXY(17,12);WRITELN('FILE DOES NOT EXIST');
            GOTOXY(17,14);WRITE('1)  TRY AGAIN');
            GOTOXY(17,15);WRITE('2)  DATA MENU');
            GOTOXY(17,19);WRITE('PLEASE SELECT NUMBER OF CHOICE: ');
            GOTOXY(49,19);
            GO:=READKEY;
            CASE GO OF
               '1' : TEST:=1;
               '2' : TESST:=3;
            END;

         END;
      END;


      IF TESST <> 3 THEN            (GIVEN THAT THE FILE EXISTS)
      BEGIN
         RESET(FIL);                                  (OPEN FIL TO BE
READ)
         FOR I:= 1 TO NUMFILE DO
```

```
        BEGIN                                          {FOR THE NUMBER
SPECIFIED}
           FOR J:=1 TO NUMENT DO



           BEGIN
              READLN(FIL,TEMP[M]);          {READ THE DATA VALUE FROM
"FIL"}
              WRITELN(DCT,TEMP[M]);            {WRITE IT INTO "DCT"}
              M:=M+1;
           END;
        END;

   CLOSE(FIL);                          {CLOSE THE FILES TO FURTHER
OPERATION}
   CLOSE(DCT);
   BOXES;

   {The user is then asked to make a choice from the menu below.  If
    the data has been loaded correctly, he/she can proceed to the
    transient identification section.  Decision between the first two
    options is based on whether there are multiple simulation output
    files or only one.  If the data was not loaded correctly, then the
    user can return to the data menu at this pt.}

   GOTOXY(17,12); WRITE('DIRECTORY ',DCTNAME,' IS NOW BUILT.');
   GOTOXY(17,13); WRITE('1) IDENTIFY TRANSIENT, MULTIPLE FILES');
   GOTOXY(17,14); WRITE('2) IDENTIFY TRANSIENT, SINGLE FILE');
   GOTOXY(17,15); WRITE('3) DATA MENU');
   GOTOXY(17,19); WRITE('PLEASE ENTER APPROPRIATE CHOICE: ');
   GO:=READKEY;
   CASE GO OF
      '1' : AVERAGE;           {GIVEN MULTIPLE OUTPUTS THIS OPTION IS
CHOSEN}
      '2' : MOVAVG;            {APPROPRIATE FOR SINGLE OUTPUT FILE}
      END;
   END;
END;



PROCEDURE RECALL;

{This procedure is used when the data to be used has been loaded
previously.
 It can be accessed either directly from the main Data Menu or through
the
 View Procedure listed below.  In either case this procedure queries the
 user for the .DCT directory to be loaded, and upon valid input recovers
```

```
 the directory.  It then prints the number of files and their length on
the
 screen for the user's information, along with providing three options.
 These options are identical to those offered in procedure Newdata
above.
 They are to Return to the Data Menu or identify the transient for
multiple
 or single files.)

VAR I,J,M,N,TEST,TESST  :INTEGER;
    FILNAME             :WRI;
    FIL,DCT             :FYLE;
    GO                  :CHAR;

BEGIN
   TEXTCOLOR(YELLOW);
   NUMFILE:=0;                    (INITIALIZE STATISTICS)
   NUMENT:=0;
   TEST:=1;
   TESST:=4;
   WHILE TEST=1 DO
   BEGIN
      CLRSCR;
      BOXES;
      TEST:=2;
      GOTOXY(30,6); WRITE('RECALL DIRECTORY ');
      GOTOXY(17,10);WRITE(' DIRECTORY TO RECALL:');READLN(DCTNAME);
      DCTNAME:=DCTNAME+'.DCT';     (ADD .DCT EXTENSION TO FILE TO RECALL)



      IF (EXIST(DCTNAME)=FALSE) THEN
      BEGIN
         GOTOXY(17,12);WRITELN('FILE DOES NOT EXIST');
         GOTOXY(17,14);WRITE('1)  TRY AGAIN');
         GOTOXY(17,15);WRITE('2)  DATA MENU');
         GOTOXY(17,19);WRITE('PLEASE SELECT NUMBER OF CHOICE: ');
         GOTOXY(49,19);
         GO:=READKEY;
         CASE GO OF
            '1' : TEST:=1;
            '2' : TESST:=3;
         END;

      END;
   END;

   IF TESST <> 3 THEN
   BEGIN
```

```
      ASSIGN(DCT,DCTNAME);          {ASSIGN STRING FOR FILE OPERATION}
      RESET(DCT);                   {OPEN THE DIRECTORY TO BE READ}

      READLN(DCT,NUMFILE);          {READ THE NUMBER OF FILES AND LENGTH}
      READLN(DCT,NUMENT);
      M:=1;

      FOR I:=1 TO NUMFILE DO
      BEGIN
         FOR J:=1 TO NUMENT DO
         BEGIN
            READLN(DCT,TEMP[M]);    {READ EACH VALUE INTO THE ARRAY
TEMP[]}
            M:=M+1;
         END;
      END;

      {Now that all the data is loaded in, the program provides the
       user with the number of files and their length to verify that
       it is the file he/she wanted.  If it is then the transient can
       be identified.  If it is not, then the user can return to the
       Data Menu and try again.}

      GOTOXY(17,10);WRITE(' NO. OF FILES IN DIRECTORY: ',NUMFILE);
      GOTOXY(17,11);WRITE(' NO. OF ENTRIES PER FILE: ',NUMENT);
      GOTOXY(17,13); WRITE('1) IDENTIFY TRANSIENT, MULTIPLE FILES');
      GOTOXY(17,14); WRITE('2) IDENTIFY TRANSIENT, SINGLE FILE');
      GOTOXY(17,15); WRITE('3) DATA MENU');
      GOTOXY(17,19);WRITE('PLEASE SELECT NUMBER OF CHOICE: ');
      GOTOXY(49,19);
      GO:=READKEY;
      CASE GO OF
      '1' : AVERAGE;        {APPROPRIATE FOR MULTIPLE OUTPUT FILES}
      '2' : MOVAVG;         {APPROPRIATE FOR SINGLE OUTPUT FILE}
      END;
   END;
END;

PROCEDURE VIEW;
{This procedure is called when the user asks for a list of the
 existing data directories.  It scans the current computer directory
 for all files with the .DCT extension and lists them on the screen.
 The user then has the option of returning to the main Data Menu or
 recalling one of the files listed through the Recall procedure.}

  VAR
    DIRINFO: SEARCHREC;
    CMD    : CHAR;

BEGIN
```

```
   CLRSCR;
   TEXTCOLOR(YELLOW);
   GOTOXY(30,6);WRITELN('LISTING OF DIRECTORIES');
   WRITELN;WRITELN;WRITELN;

   FINDFIRST('*.DCT',ARCHIVE,DIRINFO);     {LOCATES FIRST .DCT FILE}

   WHILE DOSERROR = 0 DO                   {UNTIL THERE ARE NO MORE
FILES}
   BEGIN
      WRITELN(DIRINFO.NAME);                {WRITE THE FILENAME TO
SCREEN}
      FINDNEXT(DIRINFO);                    {FIND NEXT .DCT FILE}
   END;

   GOTOXY(17,20);
   WRITE('TYPE 1 TO LOAD A FILE, 2 FOR DATA MENU: ');
   CMD:=READKEY;
   CASE CMD OF
      '1':    RECALL;           {USED TO LOAD ONE OF THE FILES LISTED}
      END;
END;


PROCEDURE DATA;
{This procedure is the starting point of the program.  When the program
 is initialized, this screen is displayed.  The user must then decide
 whether he/she wants to build a new data directory, recall one whose
 name is known, get a list of the directories which currently exist,
 or exit from the program.  Exiting from the program is always done
 from the Data Menu.  At the completion of the Batch Means and
Independent
 Replications routines the user is automatically returned to the Data
 Menu.  There are also several opportunities within the process to
return
 to this menu and start over.}

VAR GO :CHAR;

BEGIN
  GO:='3';
  WHILE GO <> '0' DO
    BEGIN
      BOXES;
      TEXTCOLOR(YELLOW);
      GOTOXY(30,6); WRITE('DATA OPERATIONS ');
      GOTOXY(17,19); WRITE('Enter number of selection: ');
      GOTOXY(17,10);WRITE('     1) BUILD NEW DATA DIRECTORY');
      GOTOXY(17,11);WRITE('     2) RECALL EXISTING DIRECTORY');
      GOTOXY(17,12);WRITE('     3) LIST EXISTING DIRECTORIES');
```

```
      GOTOXY(17,15);WRITE('     0) EXIT');
      gotoxy(44,19);
      GO:=READKEY;
      CASE GO OF
        '1':  NEWDATA;        {USED TO BUILD A NEW DIRECTORY}

        '2':  RECALL;         {USED TO RECALL A DIRECTORY WHOSE NAME IS
KNOWN}

        '3':  VIEW;           {USED TO GET A LIST OR EXISTING
DIRECTORIES}

       END;
    END;
END;

END.
```

Appendix D: Auxiliary Turbo Pascal™ Programs Used

```
PROGRAM LININT;

{THIS PROGRAM WILL PROVIDE A LINEAR INTERPOLATION OF A DATA SET,
PROVIDING EQUALLY SPACED VALUES AT INCREMENTS OF 10 TIME UNITS}

TYPE
    WRI  = STRING[20];
    FYLE = TEXT;


VAR

    TIME,NEW        :FYLE;      {Declare text files for data transfer}
    I,J,K,L         :INTEGER;
    TIM,NNEW        :WRI;       {Declare names to be associated w/files}

    ONE,TWO,Y,MAX   :REAL;
    A1,A2,B1,B2     :REAL;      {Declare coordinates to be used}
    LL              :WRI;

BEGIN

   FOR L:=1 TO 50 DO            {For the 50 SLAM II output files}
   BEGIN
      STR(L,LL);                {Define the number L as a word}

      ASSIGN(TIME,'MONE'+LL+'.OUT'); {Identify each output file in
order}
      ASSIGN(NEW,'EQUAL'+LL+'.DAT'); {Build the Lth new file}
      RESET(TIME);                   {Reset the output file to be read}
      REWRITE(NEW);             {Initialize the new file to receive
data}

      A2:=0.0;                  {Start the interpolation at the point
0,0}
      A1:=0.0;
      I:=10;                    {Initialize the observation step at 10}
      K:=1;                     {Initialize counter variable to 1}

      READ(TIME,B1);READLN(TIME,B2); {Read the data from the output
file}

      WHILE K<=1000 DO          {For the first 1000 data values}
      BEGIN
         IF (I>A1) AND (I<B1) THEN      {If I is between the two
observations}
         BEGIN
```

```
            Y:=(((B2-A2)/(B1-A1))*(I-A1))+A2; {Compute the value for I
on
                                                a line between A and B}

            I:=I+10;                          {Increment I to I+10}
            WRITELN(NEW,Y);          {Write the interpolated value to new
file}
         END;

         IF I>B1 THEN                {If I doesn't fall between A and B}
         BEGIN
            A1:=B1;                  {Reset the values of A to B}
            A2:=B2;
            READ(TIME,B1);READLN(TIME,B2); {Read the data from output
file}
            K:=K+1;                  {Increment to the next observation}
         END;
      END;

   CLOSE(NEW);CLOSE(TIME);    {After each file is done, close the files}
   END;
END.
```

```
PROGRAM CONFILE;

{This program concatenates portions of sequential data files.  Its
 purpose is to form data files of 2000 values from the equally
 spaced data files generated by LININT.PAS, for use by the
 software package FORECAST MASTER}

TYPE
    WRI = STRING[20];
    FYLE = TEXT;

VAR
    TIME,NEW      :FYLE;          {Declare file names for input/output}
    I,J           :INTEGER;       {Declare counter variables}
    TIM,NNEW      :WRI;           {Declare identifiers for filenames}
    ONE,TWO       :REAL;
    FILENAME      :WRI;
    NEWNAME,JJ    :WRI;

BEGIN
   FOR J:=1 TO 50 DO                  {For the 50 SLAM II output files}
   BEGIN
      STR(J,JJ);                      {Makes J a string (word) rather
than #}

      FILENAME:=('FOM'+JJ+'.DAT');  {Match identifiers to the Jth file}
      NEWNAME:=('K50.DAT');          {Generate the name of output file}
      ASSIGN(TIME,FILENAME);
      WRITELN(FILENAME);             {Writes filename to screen,as a
check}
      ASSIGN(NEW,NEWNAME);
      RESET(TIME);                    {Resets the output file to be read}

      APPEND(NEW);                    {Prepares the file to be appended}

      FOR I:=1 TO 460 DO
      BEGIN
         READLN(TIME,TWO);            {Simply read the first 460 values}
      END;

      FOR I:=1 TO 40 DO
      BEGIN
         READLN(TIME,TWO); {Place the last 40 values from the 50 files}
         WRITELN(NEW,TWO); {into the newfile, for a total of 2000 obs.}
      END;
      CLOSE(NEW);CLOSE(TIME);     {Close the files at end of each
iteration}
   END;
END.
```

Appendix E: Array of Time-in-System to Time-Between-Observations

T I	Time-Between-Observations																							
E	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	203	88	63	56	58	44	29	30	30	25	18	10	15	13	8	5	5	9	7	6	3	4	1	2	5
2	93	32	30	36	26	19	23	12	8	11	12	8	4	4	1	1	1	2	1	1	5	2	1	2	4
3	86	44	27	22	17	18	12	9	11	6	11	9	2	3	5	1	3	4	1	0	1	0	2	1	8
4	70	38	25	19	22	26	12	16	14	9	12	8	2	5	6	5	2	3	1	3	2	0	2	1	4
5	82	26	25	21	25	8	15	10	10	6	5	4	6	6	6	1	1	1	0	1	1	0	1	1	5
6	70	37	21	16	8	19	13	16	8	7	9	3	1	3	2	1	7	1	0	0	2	1	0	1	3
7	64	21	15	11	18	15	10	11	10	7	3	2	7	2	6	2	6	1	1	2	1	0	0	0	4
8	66	21	16	20	13	9	11	5	9	2	8	1	4	1	3	2	0	1	3	0	1	0	0	1	4
9	57	29	16	20	7	15	11	7	6	6	5	2	3	1	3	0	0	0	0	0	0	1	0	0	2
10	49	16	14	17	13	11	7	7	3	9	4	3	3	1	1	1	0	1	1	1	0	2	1	0	1
11	59	16	10	11	18	10	6	5	4	4	4	2	5	2	0	0	1	2	3	0	1	2	2	0	3
12	37	17	14	19	9	7	14	9	9	0	9	3	4	1	0	0	0	2	1	0	2	0	0	0	3
13	48	7	10	10	7	13	8	4	7	6	4	2	4	2	0	1	1	3	0	0	0	1	0	0	3
14	36	12	16	10	15	7	3	9	5	4	4	6	4	1	1	4	3	1	2	0	0	1	0	0	0
15	42	15	9	10	9	9	6	3	7	7	0	1	3	3	0	0	1	1	1	1	0	1	0	0	0
16	19	12	9	10	4	4	6	3	8	6	2	3	6	0	3	1	0	0	0	1	0	0	1	1	2
17	30	9	6	4	7	7	6	2	3	3	3	0	1	0	2	0	3	2	0	0	1	0	0	0	1
18	22	9	6	10	5	6	2	4	4	5	3	3	1	1	0	0	2	0	1	0	0	0	0	1	1
19	23	3	4	5	2	6	4	3	1	4	2	0	4	0	1	0	1	0	1	0	1	1	0	0	3
20	20	7	3	5	6	2	3	2	1	4	1	3	1	2	2	1	0	0	0	1	0	0	0	1	0
21	16	3	8	9	3	4	1	3	4	1	2	2	0	0	0	1	0	0	0	1	0	0	0	0	1
22	13	8	8	6	1	3	5	1	1	2	2	0	0	0	1	0	0	1	0	0	0	0	0	0	2
23	15	4	3	12	1	3	4	0	2	1	0	0	3	3	0	0	1	0	0	0	0	0	0	1	0
24	9	10	7	2	3	2	5	1	1	0	0	2	2	0	2	0	1	1	0	1	0	0	0	0	1
25	13	2	7	5	2	1	3	1	0	0	1	1	2	1	0	2	0	1	0	0	0	0	0	0	2
26	24	5	5	2	2	3	1	2	1	2	0	3	1	1	2	0	1	0	0	0	0	0	0	0	0
27	21	9	3	1	4	0	1	3	1	2	0	1	0	1	1	2	0	1	0	0	0	0	1	0	0
28	14	6	4	1	2	4	1	0	1	2	1	0	1	0	1	1	3	0	0	0	1	0	0	0	0
29	9	4	2	4	2	4	1	2	0	1	2	0	2	0	0	1	1	0	0	0	0	0	0	0	1
30	12	3	3	3	3	2	2	4	3	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0
31	8	1	4	3	1	2	1	2	0	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0
32	9	5	2	4	0	0	0	1	1	1	0	1	3	2	0	1	0	0	0	0	0	0	0	0	0
33	8	6	4	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	10	2	4	4	2	1	0	2	0	0	2	2	1	1	1	0	1	0	0	0	1	1	0	0	0
35	4	3	8	1	3	1	2	1	0	2	1	1	1	1	3	0	0	0	0	0	0	0	0	0	0
36	8	5	1	4	3	0	3	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	5	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	4	1	4	3	0	2	2	0	1	2	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0
39	2	1	3	1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
40	5	2	1	0	1	1	1	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
41	5	0	0	1	0	2	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
42	3	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43	4	2	1	0	2	2	2	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
44	3	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
45	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	
46	1	1	0	2	1	0	2	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
47	5	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
48	2	1	0	0	0	0	0	1	1	0	1	0	2	1	0	0	0	0	0	0	0	0	0	0	0
49	2	0	0	0	1	0	0	0	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	1
50	29	13	8	2	12	3	3	2	2	3	1	3	0	0	1	0	1	1	0	1	0	0	0	0	2

Bibliography

1. Abraham, Bovas and Johannes Ledolter. Statistical Methods for Forecasting. New York: John Wiley & Sons, 1983.

2 Borland International. Turbo Pascal 5.5 User's Guide. Scotts Valley, CA: 1989.

3. Burden, Richard L. and Douglas J. Faires. Numerical Analysis (Third Edition). Boston: PWS-Kent Publishing Company, 1981.

4. Carlson, Neal A. and Stanton H. Musick. MSOFE User's Manual: Multimode Simulation for Optimal Filter Evaluation; Oct 1987. Contract #F33615-86-C-1087, Subcontract 2013-S1, Avionics Laboratory, AFWAL/AAAN-2, Wright-Patterson AFB, OH.

5. Chen, Der-Fa R. and Andrew F. Seila. "Multivariate Inference in Stationary Simulation using Batch Means," Proceedings of the 1987 Winter Simulation Conference: 302-304, Institute for Electrical and Electronics Engineers (1988).

6. Devore, Jay L. Probability and Statistics for Engineering and the Sciences. Monterey, California: Brooks/Cole Publishing Company, 1987.

7. Diderrich, George T. "The Kalman Filter From the Perspective of Goldberger-Theil Estimators," The American Statistician, Vol. 39, No. 3, 193-198 (August 1985).

8. Fishman, G. S. "Grouping observations in digital simulation," Management Science, Vol. 24, 510-521 (January 1978).

9. Goodrich, R. L. and Stellwagen, E. A. FORECAST MASTER, Multivariate Time Series Forecasting, Scientific Systems, Inc., Cambridge MA, 1986.

10. Harrison, P. J. and C. F. Stevens. "A Bayesian Approach to Short-term Forecasting," Operational Research Quarterly, Volume 22, No. 4, 341-362 (1971).

11. Harvey, A. C. Time Series Models. New York: John Wiley & Sons, 1981.

12. Integrated Systems Inc. MATRIX, User's Guide, Version 6.0, and release notes Version 7.0; Palo Alto, CA: 1986,1988.

13. Kalman, R.E. "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering, 82, 34-45 (1960).

14. Kelton, W. David. "Random Initialization Methods in Simulation," IIE Transactions, Volume 21, No. 4, 355-367 (December 1989).

15. Kleijnen, Jack P. C. STATISTICAL TOOLS FOR SIMULATION PRACTITIONERS. New York: Marcel Dekker, Inc., 1987.

16. Maybeck, Peter S. Stochastic Models, Estimation, and Control, Vol 1, New York: Academic Press, 1979.

17. Maybeck, Peter S. Stochastic Models, Estimation, and Control, Vol 2, New York: Academic Press, 1979.

18. Mehra, Raman K. "On the Identification of Variances and Adaptive Kalman Filtering," IEEE Transactions on Automatic Control, Volume AC-15, No. 2 (April 1970).

19. Meinhold, Richard J. and Singpurwalla, Nozer D. "Understanding the Kalman Filter," The American Statistician Vol. 37, No. 2, 123-127 (May 1983).

20. Miller, Kenneth S. and Donald M. Leskiw. An Introduction To KALMAN FILTERING WITH APPLICATIONS. Malabar, Florida: Robert E. Krieger Publishing Co., 1987.

21. Neter, John et al. Applied Linear Statistical Models (Second Edition). Homewood, Illinois: IRWIN, 1985.

22. Pritsker, A. Alan B. An Introduction to Simulation and SLAM II. New York: John Wiley & Sons, 1986.

23. Reid, J. Gary. Linear System Fundamentals, Continuous and Discrete, Classic and Modern. New York: McGraw-Hill Publishing Co., 1983.

24. Ross, Sheldon M. Introduction to Probability Models. New York: Academic Press, Inc., 1985.

25. Shea, Brian L. "Maximum Likelihood Estimation of Multivariate ARMA Processes Via the Kalman Filter," Time Series Analysis, Theory and Practice 5, 1984.

26. Stein, Samuel R. Modular Intelligent Frequency, Time and Time Interval Subsystem Study Program, Research and Development Technical Report SLCET-TR-87-0717-F, Prepared by the Ball Communication Systems Division, Bloomfield, CO, January 1989.

27. Welch, Peter D. "Chapter 6, The Statistical Analysis of Simulation Results," Computer Performance Modeling Handbook. New York: Academic Press, 1983.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GOR/ENS/90M-15	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
COMPARISON OF BATCH MEANS AND INDEPENDENT REPLICATIONS TECHNIQUES TO APPLICATIONS OF THE KALMAN FILTER FOR SIMULATION OUTPUT ANLYSIS (unclassified)

12. PERSONAL AUTHOR(S)
Charles H. Porter, Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS, Thesis	FROM _____ TO _____	1990 March	132

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
20	03		KALMAN FILTERING
12	05		COMPUTER SIMULATION

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this research was twofold. First, to develop software which automated the current simulation output analysis techniques of batch means and independent replications. Second, to determine the applicability of employing Kalman filtering for simulation output analysis on the M/M/1 queue. The Output Analysis Program incorporates batch means and independent replications into a menu driven, micro-computer program. The program allows the user to quickly analyze the simulation output graphically, discard the transient and then perform the two techniques. The Kalman filter proved valuable in providing insight into the underlying system variance in steady state, and appeared to offer several avenues for further research, particularly in the area of transient identification.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL Kenneth W. Bauer, Jr., Major, USAF	22b. TELEPHONE (Include Area Code) 513-255-3362 22c. OFFICE SYMBOL AFIT/ENS